AD-A246 774

ITATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	Aug. 31, 1991	Annual July 1990 July 1991

2

4. TITLE AND SUBTITLE
Coincident Pulse Techniques for Hybrid Electronic Optical Computer Systems

5. FUNDING NUMBERS
G AFOSR-89-0469

6. AUTHOR(S)
D.M. Chiarulli, R.G. Melhem & S.P. Levitan

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Departments of Computer Science and
Electrical Engineering
University of Pittsburgh
Pittsburgh, PA 15260

8. PERFORMING ORGANIZATION REPORT NUMBER
AFOSR-TR- 92 0143

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Office of Scientific Research
Electronic & Material Sciences Dir.
AFOSR/NE Building 410 Dr. Alan Craig
Bolling Air Force Base , Washington D.C. 20332 NE

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
2305/DS

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unrestricted

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This research was an investigation of the application of coincident pulse techniques to multiprocessor interconnection networks. The research focused on three main areas: an examination of the applicability of coincident pulse techniques and required hardware to multiprocessor applications, an investigation of the limits of scalability, and an exploration of various interconnection structures which can be created using these techniques.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Electro/optical Systems Optical Computing	98
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIM.
Unclassified	Unclassified	Unclassified	

Annual Technical Report
For:

The Air Force Office of Scientific Research
Electronic and Material Sciences Directorate

Coincident Pulse Techniques for Hybrid Electronic Optical Computer Systems

Grant Number: AFOSR-89-0469

by:

Donald M. Chiarulli	Rami G. Melhem	Steven P. Levitan
Department of	Department of	Department of
Computer Science	Computer Science	Electrical Engineering
(412) 624-8839	(412) 624-8426	(412) 648-9663
don@cs.pitt.edu	melhem@cs.pitt.edu	levitan@ee.pitt.edu

University of Pittsburgh
Pittsburgh, PA 15260

Contents

1 Project Summary

This research is an investigation of the application of coincident pulse techniques to multiprocessor interconnection networks. The research focuses on three main areas: an examination of the applicability of coincident pulse techniques and required hardware to multiprocessor applications, an investigation of the limits of scalability, and an exploration of various interconnection structures which can be created using these techniques.

2 Project Objectives

Specific objectives of this research are:

- The determination of the limits placed by current technology on implementations of coincident pulse structures. These limits include, pulse width, detection limits, degree of overlap for coincidence, power distribution and pulse synchronization.
- The study of specific structures which are capable of supporting simulcasting and multicasting communications and the comparison of these structures with functionally comparable electronic systems.
- The resolution of specific configuration issues related to clock distribution mechanisms for latching data in the simulcasting structure.
- The characterization of the tradeoff between complexity and latency for choosing the number of dimensions appropriate for a particular coincident structure.
- The study of techniques for error detection and recovery in two dimensional structures.

3 Project Status

Several of the specific objectives listed above were met in the first year of our project:

- An investigation of the limits placed by current technology on the coincident pulse technique.
- Power and distribution issues for specific bus configurations.
- Investigations of specific configurations for multicasting and simulcasting.

As noted in the publications listed below, and reproduced in the appendix the second year has seen progress on the following objectives:

- The identification of the limits to scalability for these systems.

- The quantification and resolution of the "shadow problem" in linear and multi-dimensional structures.
- The generalization of the coincident structures to pipelined bus structures and the analysis of inherent advantages of pipelined communication structures for both optical and electronic interconnections.

Several of the specific objectives have been modified based on our research. In particular:

- We see a definite need for active amplification in the tapped bus structures. Therefore we are investigating the use of non-linear (Erbium doped) fiber to create a "lossless" tapped bus structure.
- The application of the signal pipelining results to reconfigurable time division multiplexed structures.

Our next set of specific goals are:

- The generalization of our work in TDM structures to more general reconfigurable optical interconnection networks
- The identification of bandwidth as a virtual resource which can be allocated dynamically on a variety of networks.
- The use of locality in source-destination address pairs to provide a mechanism of providing a dynamic reconfiguration mechanism which provides channels at optical message speed, while optimizing resources at computer algorithm speeds.

The next sections summarize some of our recent contributions, which have not yet appeared in the open literature. Other contributions are reported in the papers given in the Appendix of this report.

4. INTERCONNECTION NETWORKS REVISITED

Interconnection networks provide physical connections for communications in multiprocessor systems. Often, for technological and economical reasons, an interconnection network has limited connectivity, and connection paradigms that enhance connectivity need to be developed. The complexities of hardware and control of the network and the communication efficiency depend on both the topology of the network and the connection paradigms.

In multiprocessor systems, a processor may communicate with other processors from time to time, but not all of the time. In other words, an application program may need a set of connections, but not all of them will be used at the same time. Therefore, it may be neither feasible, nor efficient to establish all connections at all times. Instead, establishments of required connections may be interleaved such that each subset of connections is alternately established for a fixed period of time called a time slot. That is, the available bandwidth of the interconnection network may be shared among these connections in a time division multiplexed way. We call this connection paradigm *R*econfiguration with *T*ime *D*ivision *M*ultiplexing (RTDM).

4.1. RTDM IN MULTISTAGE INTERCONNECTION NETWORKS (MINs)

Let the set of the input ports and the set of the output ports of a $N \times N$ MIN be I and O respectively, where $I = O = \{0, 1, ..., N-1\}$. A path in the MIN between $i \in I$ and $j \in O$ is denoted by $p_{i \to j} = (i, j) \in I \times O$. Define a mapping, M, to be a set of paths that can be established at the same time without conflicts in the MIN. More specifically,

$M = \{p_{i \to j} \mid \text{all } p_{i \to j} \text{ can be established at the same time without conflicts, where } 0 \le i, j < N-1\}$

Note that, an *admissible* (or permissible) permutation is a mapping that contains N paths. We will refer to mappings that contain less than N paths as *partial* mappings. Since establishment of two paths at the same time may cause conflicts, not every set of paths is a mapping. We refer to the establishment of all the paths in a mapping as the *realization* of the mapping.

Given a mapping, there is a way to set switches in the MIN to realize the mapping. Let $m = \frac{N}{2}$ be the number of switches per stage, and let $n = \log N$ be the number of stages in the MIN. Define a *switch setting* array to be an $m \times n$ array, whose i-th element at j-th column corresponds to the i-th switch at j-th stage in the MIN. Denote the switch setting array of a mapping M by $SS(M)$ and its elements by $SS(M)<i, j>$ where $1 \le i \le m$ and $1 \le j \le n$. The value of an element in $SS(M)$ is "0" or "1" if the corresponding switch has to be set to "straight" or "cross" to realize mapping M. The value of an element is "×" if the corresponding switch can be in any state without affecting the realization of the mapping, in which case, the mapping must be a partial mapping. Two mappings M_1 and M_2, are said to be not *compatible* with each other, if there are some i and j such that the two elements, $SS(M_1)<i, j>$ and $SS(M_2)<i, j>$, are either "0" or "1" but not equal. That is, M_1 and M_2 are not compatible if the

realization of both mappings at the same time will cause conflicts in switch setting. Otherwise, M_1 and M_2 are said to be *compatible* with each other, in which case, the two mappings can be merged into one mapping, namely $M = M_1 \cup M_2$.

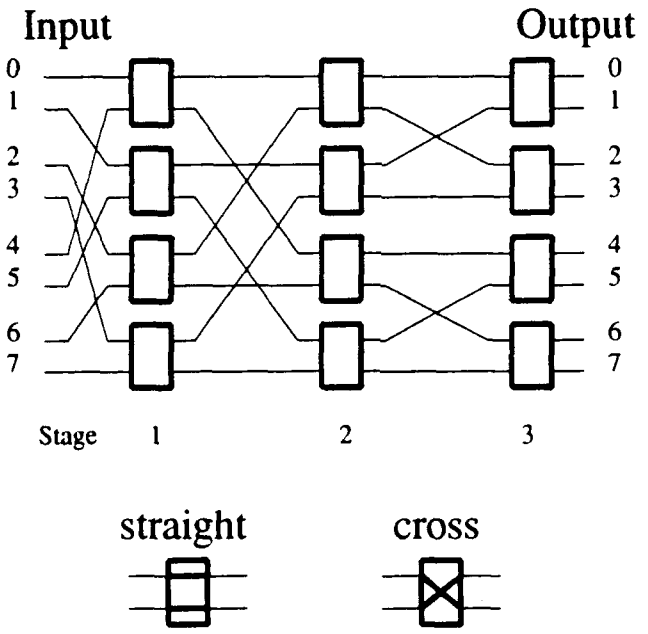


Figure 1. An 8×8 generalized cube network

MINs under consideration in this report are generalized cube networks, which are topologically equivalent to many blocking MINs. An $N \times N$ generalized cube network has n (where $N = 2^n$) stages of 2×2 switches with cube-type connections. Figure 1 shows an example of such a MIN with $N = 8$ in which stages are numbered 1 to $n = 3$ from left to right. Each switch is assumed to have two states : straight or cross, as shown in the figure. Three switch control strategies are possible for this type of MINs. Individual switch control assumes one control signal per switch. Individual stage control assumes one control signal per stage and partial stage control assumes $i+1$ control signals in stage i. In general, individual switch control is used since it yields more powerful connectivities in a MIN.

Given a set of paths $P \subseteq I \times O$, it may not be possible to establish all paths in P at the same time without conflicts. However, P can be partitioned into several mappings, $P = M_1 \cup M_2 \cup \cdots \cup M_t$. Each mapping M_i, $i = 1, 2, ..., t$, may be realized for a fixed length of time, which we call a time slot. By doing so, every path in P is established once in a time slot and P is said to be realized through time-division multiplexing. Note that, the switch setting arrays for different mappings are usually different. Therefore, the MIN has to change its switch setting after each time slot.

We call a MIN a *Time-Division Multiplexed MIN* (TDM-MIN) if it repeatedly realizes a sequence of mappings in a time-division multiplexed way. More specifically, a t-way TDM-MIN changes its switch setting after each time slot to realize one of t mappings $M_1, M_2, ..., M_t$ in a round-robin fashion. Without loss of generality, we assume that M_i is realized during the i-th time slot ($1 \leq i \leq t$). We call the ordered sequence $[M_1, M_2, ..., M_t]$ a *configuration* of the t-way TDM-MIN and the number of mappings in the sequence, t, the *Multiplexing Cycle Length* (MCL) of the configuration. Note that, this definition of configuration of a MIN is different from the conventional one that defines a configuration of a MIN as a set of numberings of input and output ports within a given MIN topology.

Once a configuration of a TDM-MIN has been determined, each mapping and its corresponding switch setting array in each time slot are determined. That is, in each time slot, the output port to which any given input port is connected is known. So is the state to which any given switch should be set. A global clock may be used to synchronize all input ports and switches in the MIN at the beginning of each time slot. Each input port maintains a list of output ports to which it is connected during different time slots of a multiplexing cycle. More specifically, the k-th entry in the list of source node i is j if $p_{i \rightarrow j} \in M_k$. Each switch is assumed to have a shift register whose size is no less than the multiplexing cycle length, t, of a configuration. The sequence of t states that a switch should be set to is stored in the shift register. The k-th bit of the shift register of the i-th switch at stage j is either "0" or "1" if the corresponding element of $SS(M_k)$ is "×". Otherwise, it should be equal to the value of its corresponding element of $SS(M_k)$.

At the beginning of each time slot, every switch is set to the state specified by the content of its shift register. After switches are set properly, an input port can transmit a message to the output port to which it is connected in this time slot. Note that, if individual stage control is used, only one shift register per stage is required.

In MIMD environments, a MIN is commonly either circuit-switched or packet-switched or a combination of both. In circuit-switching, only a limited number of circuits can be established without conflicts. Dynamically setting up or releasing a circuit involves run time overheads. In packet switching, having one routing tag for each packet also introduces run time overheads. In addition, switches are more complex since they need to do buffering and arbitrations based on routing tags of incoming packets.

In a TDM-MIN with multiplexing cycle length t, up to tN different connections can be established. If a set of connections required by an application is known, a MIN can be set to a TDM configuration statically. This means that after execution begins, the time slot in which each connection is established is predetermined and routing decisions are as simple as waiting for the appropriate time slots. As a result, messages do not have to contain routing information such as destination addresses, nor do they need to be buffered at any intermediate switch. Overheads due to arbitration and path conflicts in circuit or packet switchings are eliminated at run time.

Even in applications that require dynamically changes of connections, the overall communication pattern is expected to change slowly during execution. Dynamic reconfiguration may affect one or more mappings but most of the other mappings will remain intact. As a result, overheads due to dynamic establishments or releases of connections are relatively lower than using circuit switching. In essence, the RTDM connection paradigm takes advantage of the relative stability of communication patterns to simplify control and to reduce overheads. On the other hand, however, the multiplexing degree in a TDM-MIN affects the latency of a connection, which must be kept low to achieve high communication efficiency.

4.2. STATIC RECONFIGURATIONS

4.2.1. Connection Request Graphs

Communication requirements of an application can often be obtained as a result of compile time analysis. After data allocation and processor assignment are done, memory access patterns or inter-processor communication patterns can be represented by a bipartite graph, which we call *Connection Request* (or CR) graph.

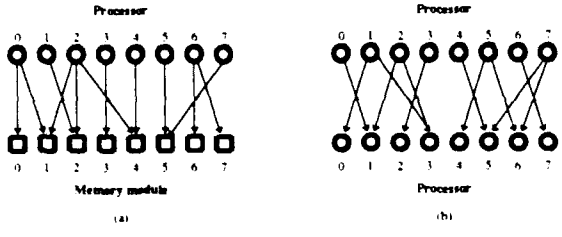


Figure 2. Examples of Connection Request (CR) Graphes

Figure 2 shows two examples of CR graphs, one for shared memory systems and another for message passing systems. Figure 2(a) shows a CR graph based on processor to memory connections. A directed edge from processor i at the top to memory module j at the bottom means that processor i may address memory module j. On the other hand, Figure 2(b) shows a CR graph based on inter-processor connections. A directed edge from a source processor i at the top to a destination processor j at the bottom means that processor i may send messages to processor j. An edge from processor i to itself is meaningless in CR graphs for interprocessor connections. Note that, due to the dynamic nature of memory access requests in many applications, it is relatively difficult to construct CR graphs for shared memory systems.

A node in a CR graph is called either a *source node* or a *destination node*. Let source node i, $0 \le i \le N-1$, use input port i of an $N \times N$ MIN and let destination node j, $0 \le j \le N-1$, use output port j of the same MIN. Therefore, an edge from source node i to destination node j in a CR graph requires the establishment of a path $p_{i \to j}$ in the MIN. We will use the same notation for a path to denote an edge and use the terms "edge" and "path" interchangeably.

Denote the set of all edges in the CR graph by E, and the number of edges in the set by $|E|$. As an example, the set E in the CR graph in Figure 2(b), with $|E| = 12$, is given in Eq. 3.1 below.

$$E = \{(0,1), (1,0), (1,3), (2,1),(2,3), (3,2), (4,5), (5,4), (5,6), (6,7), (7,5), (7,6)\} \quad (3.1)$$

Note that, any directed or undirected communication graph can be converted into a corresponding bipartite CR graph. The number of edges going out from a node is called the *out-degree* of the node and the number of edges coming into a node is called the *in-degree* of the node. We will refer the maximum of the out-degree and the in-degree of a node as the *degree* of a node.

Given a CR graph, we call a configuration $[M_1, M_2, ..., M_t]$ a *Minimal Connection* (or MC) configuration for the CR graph if it satisfies the following two conditions.

(1). $E \subseteq \bigcup_{i=1}^{t} M_i$.

(2). for any $i, j \in \{1, 2, ..., t\}$, M_i and M_j are not compatible.

The first condition states that any edge $(i, j) \in E$ is established in a mapping M_i and the second condition states that any two mappings in the configuration can not be merged together. We call a configuration for a CR graph *optimal* if it is an MC configuration for the graph and it has the least multiplexing cycle length among all other MC configurations for the same graph. Note that, if t is equal to the maximum degrees of nodes in a CR graph, then the MC configuration is optimal.

4.2.2. Embeddings of Regular Communication Structures

When the communication structure of an application is regular, finding a configuration for its CR graph is often called *embedding*. Note that, the ability to embed regular communication structures efficiently is important since there are many existing applications designed for them. The multiplexing cycle length t of a configuration is a measure of the efficiency of the embedding. This measure is, in some sense, similar to the dilation cost in conventional embeddings. We also define *path utilization*(PU) to be the ratio of the number of connections required versus the number of connections that can be established in one multiplexing cycle. That is, $PU = \frac{|E|}{Nt}$.

Since there are N^2 paths between every input port and every output port in a completely connected CR graph and at most N paths can be established in each mapping, an MC configuration that embeds a completely connected network has at least N different mappings. We call a configuration $[M_1, M_2, ..., M_N]$ such that

$$\bigcup_{i=1}^{N} M_i = I \times O \tag{3.2}$$

a *completely connected* (CC) configuration since every path in a completely connected network is established in one of the N mappings of the configuration. Such an embedding is clearly an optimal one with its multiplexing cycle length $t = N$ and path utilization $PU = 1$. There are more than one CC configurations. As one example, define

$$M_{s(k)} = \{p_{i \to j} \mid j = (i + k) \bmod N \quad \text{for } i = 0, 1, ..., N-1\} \tag{3.3}$$

and call it a *shift-k* mapping. Therefore, the configuration $[M_{s(0)}, M_{s(1)}, ..., M_{s(N-1)}]$ establishes paths from any input port to all N output ports and, thus, is a CC configuration. As another example, define

$$M_{f(k)} = \{p_{i \to j} \mid j = i \text{ xor } k \quad \text{for } i = 0, 1, ..., N-1\} \tag{3.4}$$

and call it a *flip-k* mapping where *xor* is the bit-wise *Exclusive-OR* operation. The *flip-k* mapping can be realized by individual stage control. Therefore the configuration $[M_{f(0)}, M_{f(1)}, ..., M_{f(N-1)}]$ is also a CC configuration. Note that, CC configurations are functionally equivalent in terms of their multiplexing cycle lengthes and path utilizations. However, the CC configuration with *flip-k* mappings may be chosen for the purpose of embedding a completely connected network in the time domain due to its control simplicity. It is also worth noting that in the case of processor-to-processor interconnection, the *identity* mappings, such as $M_{s(0)}$ or $M_{f(0)}$, that establish no paths other than those from a node to itself can be deleted from the CC configurations.

Since any CR graph is a subgraph of a completely connected graph, it can be embedded in a CC configuration of a TDM-MIN. This, however, requires an N-way TDM-MIN and thus may be inefficient in terms of both the multiplexing cycle length and the path utilization. An alternative is to find an MC configuration of length $t < N$ which embeds the CR graph. In other words, a t-way TDM-MIN instead of an N-way TDM-MIN, for some $t < N$, can be used to increase the embedding efficiency. The smaller the ratio of $\frac{t}{N}$ is, the more efficient it is to use such an MC configuration. Table 1 summarizes embedding results of several regular communication structures.

4.2.3. Static Reconfigurations Based On Arbitrary CR Graphs

For non-regular CR graphs, an MC configuration can always be obtained by selecting a subset of mappings from a CC configuration. Such an MC configuration can often improve performance of applications by reducing the multiplexing cycle length to t time slots, for some $t < N$. Note that, given a specific path, there is only one mapping in a CC configuration that establishes that path. The mapping can usually be determined by either a simple arithmetic operation or a table look-up. For example, if the CC configuration consists of flip-k mappings, the mapping that will establish the path $p_{i \to j}$ is $M_{f(k)}$ where $k = i \text{ xor } j$.

Structure	Nodes	MCL	Optimal
Ring	N	2	yes
Mesh	$N=m^2$	4	yes
Hypercube	$N=2^n$	n	yes
Cube-Connected Cycle	$N=2^n$	3	yes
Complete Binary Tree	$N=2^n-1$	4	?

Table 1. Embedding Regular Structures in TDM-MINs

Given a CR graph containing a set of edges E and a CC configuration $[M_1, M_2, ..., M_N]$, an MC configuration that establishes the edges in the CR graph can be found by using the selection algorithm below. We use the symbol [ ] to denote an empty MC configuration with no mappings and the operation || to denote the addition of a mapping to an MC configuration.

Selection Algorithm

1. Set initially MC = [ ]
2. For each edge $p_{i\rightarrow j} \in E$ repeat
   2.1. Determine the mapping M_k such that $p_{i\rightarrow j} \in M_k$
   2.2. If $M_k \notin$ MC then MC = MC || M_k

For example, consider the CR graph in Figure 2(b) and the CC configuration consisting of *flip-k* mappings. The MC configuration selected by the algorithm is $[M_{f(1)}, M_{f(2)}, M_{f(3)}]$. Since $t = 3$ and $N = 8$, a 3-way rather than an 8-way TDM-MIN may be used for the application to improve the efficiency. Note that, the maximum number in the set of in-degrees and out-degrees of nodes in the graph is 2 and, thus, this configuration may not be optimal. In fact, an optimal MC configuration with $t = 2$ will be obtained in the next section.

Probabilistic analysis of the average multiplexing cycle length of MC configurations for random CR graphs can be carried out as follows. Given a CR graph in which S out of N source nodes are each connected randomly to D out of N destination nodes, define $s = \frac{S}{N}$ and $d = \frac{D}{N}$. The probability that an edge is established in any mapping of a given CC configuration is $\frac{1}{N}$. Since edges that go out from the same source node must be established in different mappings, exactly D mappings are needed to establish paths from a source node to D destination nodes. Therefore, after selecting D mappings from the given

CC configuration, the probability that any mapping has *not* been selected is $p = 1 - d$. Since each source node randomly selects D mappings independent of others, the probability that a mapping in the CC configuration remains un-selected after all S source nodes have selected their mappings is $P = p^S$. The probability that exactly i mappings have been selected for an MC configuration is thus

$$Prob(i) = \binom{N}{i} P^{N-i}(1-P)^i \tag{3.5}$$

Therefore, the average (expected) number of mappings selected, that is, the expected multiplexing cycle length of a MC configuration is

$$t_{av} = \sum_{i=d}^{N} i \times Prob(i) \tag{3.6}$$

Clearly, $t_{av} \geq S$, since S is the out-degree of a source node. The percentage of communication load of an application can be approximated by the ratio of $|E|$ versus N^2. In the above analysis, the load percentage is proportional to $s \times d$. Figure 3 shows calculated values of $\frac{t_{av}}{N}$ for different system size N under different load conditions assuming $s = d$. It can be seen that with reasonably small system size and under low load condition, the selection algorithm can generate an MC configuration that improves over a CC configuration. Note that, by using the CC configuration with $flip-k$ mappings, the selection algorithm can be simple and so does the resulting MC configuration because of individual stage control. The time complexity of the algorithm is linear in the number of connections requested, that is $O(|E|)$.

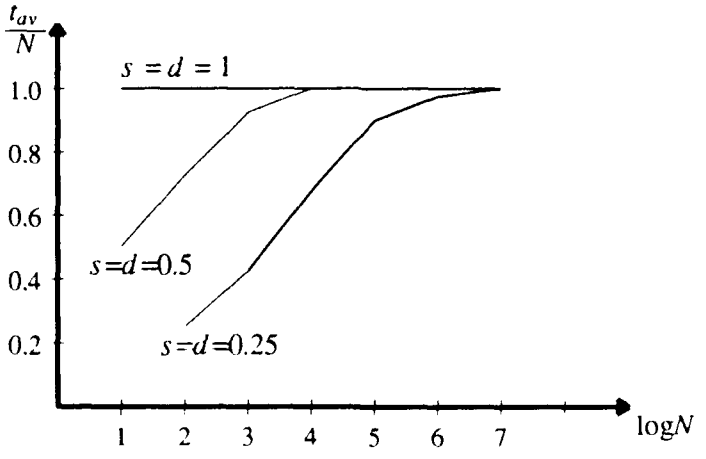


Figure 3. Percentage of mappings selected

The problem with the selection algorithm is that it restricts itself to the set of N mappings in a chosen CC configuration. With individual switch control, however, a total of $2^{mn} = N\sqrt{2^N}$ mappings (excluding partial mappings) are possible in the MIN. Given a CR graph with a set of edges E, a MC configuration can be obtained by composing mappings based on the set E, which may be different from

any mapping in a chosen CC configuration. More specifically, assuming individual switch control, the following *composition* algorithm composes each mapping in a greedy fashion. That is, starting with an empty set of paths, a mapping is composed by including as many required paths as possible provided that they do no conflict.

The Composition Algorithm

Set MC = [ ] and $k = 1$. Repeat until E is empty

1. Reset mapping $M_k = \phi$ and all elements of $SS(M_k)$ to "×"

2. For each edge $p_{i \to j} \in E$

   If $\{p_{i \to j}\}$ is compatible with M_k

   2.1. $M_k = M_k \bigcup \{p_{i \to j}\}$ and update $SS(M_k)$ accordingly

   2.2. delete $p_{i \to j}$ from the set E.

3. MC = MC || M_k and $k = k + 1$

For example, an MC configuration that is composed by the algorithm for the CR graph in Figure 2(b) is $[M_1, M_2]$. For easy verifications by the readers, we will show each mapping in the MC configuration with its corresponding switch setting array in Eq. 3.7.

$$M_1 = \{(0,1), (1,0), (2,3), (3,2), (4,5), (5,4), (6,7), (7,6)\}$$

$$M_2 = \{(1,3), (2,1), (5,6), (7,5)\} \tag{3.7a}$$

Their switching setting arrays are respectively:

$$SS(M_1) = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \qquad SS(M_2) = \begin{bmatrix} 0 & 1 & 1 \\ 0 & \times & 1 \\ 1 & \times & 0 \\ 0 & 1 & 1 \end{bmatrix} \tag{3.7b}$$

In this example, the MC configuration composed by the algorithm is optimal with $t = 2$, which improves over the MC configuration generated by the selection algorithm. Simulations have been done to determine the average multiplexing cycle length of MC configurations composed by the composition algorithm. A random number generator is used to generate D distinct destination nodes for each of the S source nodes. Figure 4 shows simulation results of t_{av} for different system sizes under different load conditions where $s = \frac{S}{N}$ is equal to $d = \frac{D}{N}$. It can be seen that under low or medium load conditions, the composition algorithm improves over the selection algorithm as expected. However, when the load is extremely high, the multiplexing cycle length of an MC configuration could exceed N. That is, configurations under high load condition using this algorithm may be worse than simply using a CC configuration. Note that, $\frac{t_{av}}{N}$ does not vary much with the system size N. Figure 5 shows simulation results for a system with $N = 32$ with various s and d. Given that it takes $O(logN)$ time to compute switch settings for a path, the composition algorithm has the time complexity of $O(|E|^2 logN)$.

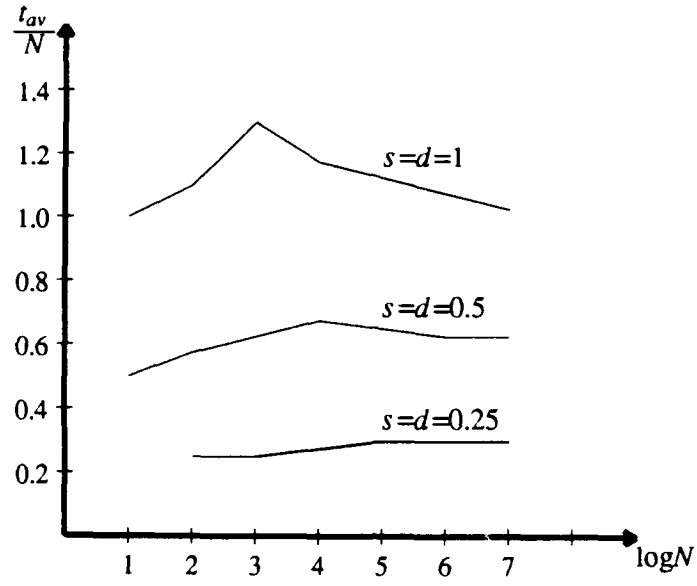


Figure 4. Percentage of mappings composed

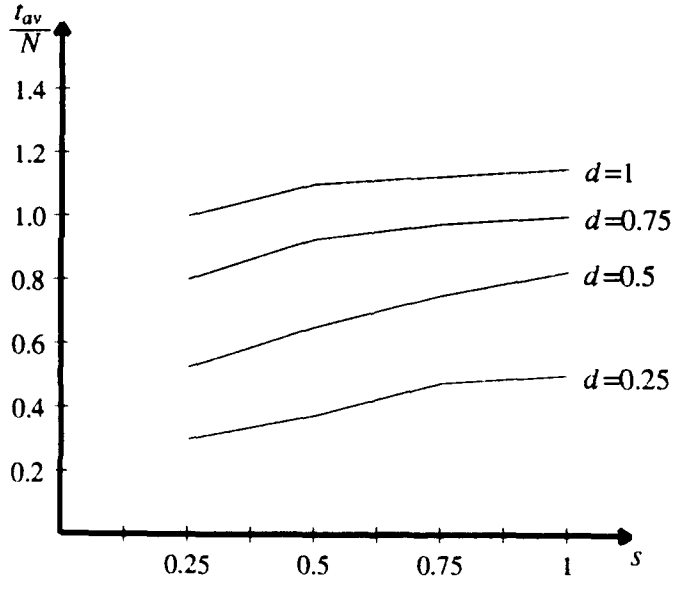


Figure 5. Different load conditions when $N = 32$

It is desirable, not only to do better than the selection algorithm under low or medium load conditions, but also to bound the multiplexing cycle length of any MC configuration by N under high load conditions. One way is to use the selection algorithm first to determine a set of up to N $flip$-k mappings needed. Then each mapping is examined to see if it can be deleted from the configuration by *migrating*

paths established in it to other mappings in the configuration. Given a CR graph with a set E, we can use the following merge algorithm to achieve the above objective.

The Merge Algorithm

1. Run the selection algorithm to determine an MC configuration.

2. For each mapping $M_k \in$ MC, repeat step 3

3. If every $p_{i\rightarrow j} \in M_k$ is such that $\{p_{i\rightarrow j}\}$ is compatible with a M_l currently in MC where $l \neq k$

   3.1. For every $p_{i\rightarrow j} \in M_k$

   $M_l = M_l \cup \{ p_{i\rightarrow j} \}$ if $\{ p_{i\rightarrow j} \}$ is compatible with M_l

   3.2 Remove M_k from MC

Simulations have been done under similar assumptions to those used for the composition algorithm. Figure 6 shows the results of the merge algorithm for a system with $N = 32$. It can be seen that the merge algorithm performs as good as the composition algorithm under low or medium load conditions and converges to the selection algorithm under high load conditions. The complexity of this algorithm can be shown to be $O(N \mid E \mid^2 \log N)$.

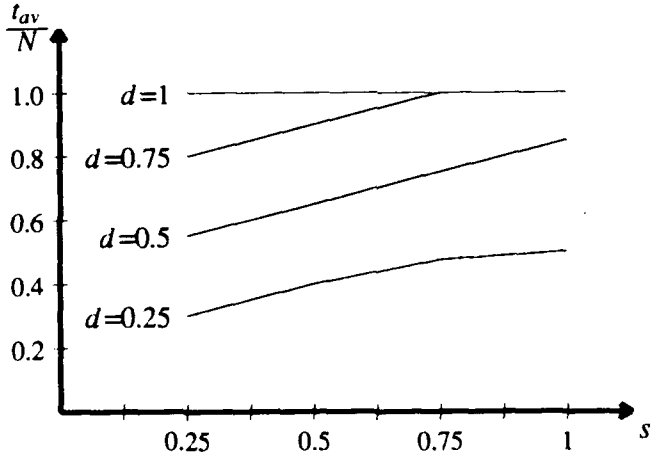


Figure 6. Performance of the merge algorithm

4.3. DYNAMIC RECONFIGURATIONS

Static reconfigurations work well if all paths are required to be established from the beginning to the end of executions of an application. For applications such as those in telecommunication, connection requests are usually generated at run time. Even if a CR graph that contains all edges needed during the execution can be constructed at compile time, it may be inefficient to perform static reconfigurations based on such graph since some paths are used only for a certain duration of time and are wasted for the

remaining time during execution. It is possible to achieve more efficient communication in such an application by reconfiguring a TDM-MIN dynamically based on run time requests. This means, mappings realized in a time slot may be different from time to time. We call such reconfigurations *dynamic reconfigurations.*

4.3.1. Centralized Reconfigurations

Run time requests may include requests for establishing new paths and releasing existing ones. Dynamic reconfigurations can be done incrementally based on an existing configuration. If a request is to establish a path $p_{i\rightarrow j}$, the current configuration is examined to find a mapping M_k that is compatible with $\{p_{i\rightarrow j}\}$. If successful, path $p_{i\rightarrow j}$ is added to mapping M_k by updating $SS(M_k)$. In the array, only elements that correspond to switches along the new path and have value of "×" are updated to either "0" or "1". Consequently, the k-th bit of each shift register of those switches whose corresponding elements have changed values may need to be updated. The source node i also sets its k-th entry in the list of output ports to j. At this time, reconfiguration based on the request is completed.

If, however, the current configuration does not contain any mapping that is compatible with $\{p_{i\rightarrow j}\}$, a new mapping that establishes $p_{i\rightarrow j}$ can be added to the configuration. This requires that all source nodes be informed of the additional time slot in the multiplexing cycle. Shift registers of all switches have to be updated accordingly. Before adding the new mapping, one can migrate existing paths in a mapping into other mappings so that it may become compatible with $\{p_{i\rightarrow j}\}$. This way, the new mapping may be avoided. Note that there are tradeoffs between overheads of migrating paths and overheads of adding a new mapping.

If a run time request is to release a path, the mapping that currently establishes the path may be deleted if all remaining paths in the mapping can be migrated into other mappings in the configuration. Such explicit release requests may not be necessary if replacement algorithms or garbage collection algorithms are used by the central controller based on the usage of existing connections. All these involves tradeoffs. Note that, dynamic reconfiguration can also be done by buffering run time requests and periodically executing a static reconfiguration algorithm. At each selected instance, a snapshot of the CR graph is constructed based on all current paths that need to be established.

4.3.2. Distributed Reconfiguration

Assume that nodes connected to a MIN has distributed control but global synchronization is still applicable. In this case, the multiplexing cycle should always consist of a fixed number, k, of time slots. This is because each node will not be aware of either increment or decrement of the number of time slots in the multiplexing cycle (in a timely way) without being informed by a centralized control mechanism.

Each source node that wants to establish a path reserves a time slot in which the path may be established by routing a reservation packet to the destination on the path. These reservation packets use links and switches, called *reservation links* and *reservation switches* respectively, separate from those used by data packets. Note that the separation can be either *logical* or *physical*. An example of logical separation could be a MIN with links and switched used by reservation packets and data packets in a time-multiplexed way. In the following discussion, terms "link" and "switches" are used to refer to reservation links and reservation switches respectively.

Let an input port of a switch s be denoted by $l(s)$ and let an output port of the switch be denoted by $r(s)$. Let a path be represented by a sequence of $n = \log N$ pairs of left and right ports of switches at each stage. That is, $p_{i \to j}$ can be represented by $(<l(s_1), r(s_1)>, <l(s_2), r(s_2), ..., <l(s_n), r(s_n)>)$. Note that, this implies that $r(s_i)$ is connected to $l(s_{i+1})$. Let every output port $r(X)$ maintain a set of time slots that is not used by any paths. Denote that set by $AVAL(r(X))$. Assume that each source node also maintains an $AVAL(l(Y))$ list for an input port $l(Y)$ to which it is connected. Let "lock" and "unlock" be mutual exclusive operations on a switch port. Only the reservation packet that can successfully "lock" the port can update its $AVAL$ list while other reservation packets are buffered at the switch until the port is "unlocked". When the TDM-MIN system is started, all ports are unlocked and for any port Z, $AVAL(Z) = \{1, 2, ..., k\}$.

Each reservation packet maintains a set of time slots that are available for possibly establishing the corresponding path. Denote by $AVAL(R)$ the set of available time slots maintained by a reservation packet R. When a reservation packet R is generated, its $AVAL(R)$ is set to $AVAL(l(s_1))$. As a reservation packet goes through each switch, it locks the corresponding ports and updates its own $AVAL(R)$ to the set of time slots that are available at all switch ports visited so far. If the reservation packet reaches the destination, it chooses a time slot, namely $ts \in AVAL(R)$, and returns to the source along the same path in reverse order. As it passes each switch, it deletes the ts from the $AVAL$ lists of each port visited and unlocks these ports. At the same time, the ts-th bit of the shift register of the switch is loaded with a proper state. When it comes back to the source node, the destination to which it is sent to is recorded in the ts-th entry of the list of output ports by the source node.

Before the control packet reaches its destination, if $AVAL(R)$ would become empty at a switch, the reservation packet may be blocked. Two strategies similar to "holding" and "dropping" in circuit switching can be used when a packet is blocked. If holding is used, the packet stays in the buffer of the switch. An advantage is that whenever some paths using the same switch port are released, the reservation packet can continue its routing without repeating from the source up to that switch. However, a disadvantage is that switch ports that have been locked by the packet can not be used by other reservation packets while the packet is blocked. An alternative is to use dropping, in which the reservation packet reverses its way, unlocking switch ports and undoing changes to $AVAL$ sets of switches. The source node may queue the packet and try to send the packet again after a random interval. Note that, a combination of these two

strategies, which drops a packet after holding it for a certain period, can also be used.

If a source node wants to release a path, it sends a cancellation packet $\bar{R}$ with $AVAL(\bar{R})$ containing the time slot in which the path is established. The cancellation packet can add the time slot in $AVAL(\bar{R})$ into $AVAL$ sets of every switch ports visited on the way to its destination. Assume that dropping is used, then the following algorithm may be executed distributively when establishing and releasing a path.

The Distributed Algorithm (with dropping)

Establish($(<l(s_1), r(s_1)>, <l(s_2), r(s_2), ..., <l(s_n), r(s_n)>)$)

1. The source node connected to $l(s_1)$ generates a reservation packet R with $AVAL(R) = AVAL(l(s_1))$

2. For $i = 1$ to n do

    2.1. Lock port $r(s_i)$, $AVAL(R) = AVAL(R) \cap AVAL(r(s_i))$

    2.2. If $AVAL(R) = \phi$ then

        2.2.1. For $j = i$ downto 1 do unlock port $r(s_j)$

        2.2.2. The source node realizes the path is not established

        2.2.3. Exit this procedure

3. Choose a time slot $ts \in AVAL(R)$

4. For $i = n$ downto 1 do

    4.1 $AVAL(r(s_i)) = AVAL(r(s_i)) - \{ts\}$ and unlock port $r(s_i)$.

5 $AVAL(l(s_1)) = AVAL(l(s_1)) - \{ts\}$.

Release($(<l(s_1), r(s_1)>, <l(s_2), r(s_2), ..., <l(s_n), r(s_n)>)$)

1. The source node connected to $l(s_1)$ generates a cancellation packet $\bar{R}$ with $AVAL(\bar{R}) = \{ts\}$

2. For $i = 1$ to n do

    2.1. Lock port $r(s_i)$, $AVAL(r(s_i)) = AVAL(r(s_i)) \cup \{ts\}$. Unlock port $r(s_i)$

3. $AVAL(l(s_1)) = AVAL(l(s_1)) \cup \{ts\}$

Note that, when the queue containing unsent packets is not empty, a source node may periodically execute the procedure *Establish()* from step 2. An algorithm with holding can be similarly written, so does an algorithm with the combination of holding and dropping.

4.4. SUMMARY

To summarize, reconfiguration with TDM is a connection paradigm that can be applied to multiprocessor systems using multistage interconnection networks. It provides more architectural flexibilities and could achieve potentially higher communication bandwidthes than conventional switching methods. It is especially promising for optical interconnection networks because, first, high optical communication bandwidths make time-division multiplexing feasible and more attractive; Important properties of optical signal propagation, namely unidirectional propagation and predictable path delay, enable pipelined transmissions over optical waveguides. The way in which switches are set in TDM-MIN can further simplify pipelinings between stages. Second, partitioning connection requests and establishing subsets in a time division multiplexed way can simplify controls and eliminate the needs for message relaying, optical delay loops (optical time-slot interchangings) and costly conversions between optical and electronic signals. As a result, current photonic switching technology, can be readily adopted for TDM-MINs implementations.

5 Project Publications: Previous, Current and in Preparation

Publications in Preparation

- "Efficient channel allocation for routing in optically interconnected multiprocessor systems"; C.Qiao, R.Melhem, D. M. Chiarulli, S. P. Levitan; SPIE Symposium on OE/Aerospace Sensing'92, Conf. on Advances in Optical Information Processing V; Orlando, Fl.; April 20-24, 1992.
- "Bandwidth as a virtual resource"; D. M. Chiarulli, S. P. Levitan, R. G. Melhem (in preparation).

Current Publications (this funding period)

- *Array processors with pipelined busses and their implication in optically and electronically interconnected multiprocessors*; Zicheng Guo; Ph.D. thesis; Department of Electrical Engineering, University of Pittsburgh, 1991.
- "Demonstration of an all optical addressing circuit"; D. Chiarulli, S. Levitan, R. Melhem; Optical Society of America Topical Meeting on Optical Computing; Technical Digest Vol. 6, TuC3-1, pp.235-238; Salt Lake City, UT, March 4-6, 1991.
- "An all optical addressing circuit: experimental results and scalability analysis"; Donald M. Chiarulli, Robert M. Ditmore, Steven P. Levitan, and Rami G. Melhem;*IEEE Journal of Lightwave Technology, Vol. 9*, No. 12, pp. 1717-1725, 1991.
- "Optical multicasting in linear arrays"; Chunming Qiao, R.G. Melhem, S.P. Levitan and D.M. Chiarulli,(in press) *International Journal on Optical Computing.*
- "Pipelined communications in optically interconnected arrays"; Z. Guo, R.G. Melhem, R.W. Hall, D.M. Chiarulli, and S.P. Levitan; *Journal of Parallel and Distributed Computing, Vol. 12*, No. 3, pp. 269-282, 1991.
- "Multicasting in optical bus connected processors using coincident pulse techniques"; Chunming Qiao, R. Melhem, D. Chiarulli and S. Levitan; International Conference on Parallel Processing; (poster); St. Charles, IL, August 20-23, 1991.
- "Time-division optical communications in multiprocessor arrays"; C. Qiao, R. Melhem; Proc. of the Supercomputing 91 Conference, Albuquerque, NM, November, 1991; IEEE Press (1991).

Previous (Related) Publications

- "Self routing interconnection structures using coincident pulse techniques;" D.M. Chiarulli, S.P. Levitan, and R.G. Melhem; In *SPIE OE/Boston'90*, 1390-25, S4, pp. 403-414; Boston, MA, November 4-9 1990.

- "Pipelined communications on optical busses"; Z. Guo, R. Melhem, R. Hall, D. Chiarulli, and S. Levitan; In *SPIE OE/Boston'90*, 1390-26, S4, pp. 415-426; Boston, MA, November 4-9 1990.
- "Embedding pyramids in array processors with Pipelined busses"; Zicheng Guo and Rami Melhem; In *Intl. Conf. on Application Specific Array Processors*, pages 665–676, Princeton, N.J., 1990.
- "Array processors with pipelined optical busses"; In *Frontiers'90: 3rd Symposium on the Frontiers of Massively Parallel Computation*, Z. Guo, R. Melhem, R. Hall, D. Chiarulli, and S. Levitan; pp.333-324; University of Maryland College Park, MD, October 8-10 1990.
- "Coincident pulse techniques for multiprocessor interconnection structures"; S.P. Levitan, D.M. Chiarulli, and R.G. Melhem; *Applied Optics*, 29(14):2024–2033, May 10 1990.
- "Optical bus control for distributed multiprocessors"; D.M. Chiarulli, S.P. Levitan, and R.G. Melhem; *Journal of Parallel and Distributed Computing*, 10:45–54, 1990.
- R.G. Melhem, D.M. Chiarulli, and S.P. Levitan; "Space multiplexing of waveguides in optically interconnected multiprocessor systems", *The Computer Journal, British Computer Society*, 32(4):362–369, 1989.
- Donald M. Chiarulli, Rami Melhem, and Steven P. Levitan. Asynchronous control of optical busses in closely coupled distributed systems. Technical Report 88-2, Department of Computer Science, University of Pittsburgh, 1988.
- D.M. Chiarulli, R.G. Melhem, and S.P. Levitan. "Using coincident optical pulses for parallel memory addressing", *IEEE Computer*, 20(12):48–57, December 1987.

6 Project Personnel

6.1 Current Vita of Principal Investigators

- Donald M. Chiarulli, Co-PI
- Rami G. Melhem, Co-PI
- Steven P. Levitan, Co-PI

Curriculum Vitae

Donald M. Chiarulli

Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260
412-624-8839
INTERNET: don@cs.pitt.edu

Research Interests

Algorithms and architectures for the design and implementation of highly parallel computing systems are the focus of my research.

Education

Ph. D. in Computer Science, 1986, Louisiana State University, Baton Rouge.
Dissertation: "A Horizontally Reconfigurable Architecture for Extended Precision Integer Arithmetic."

M. S. in Computer Science, 1979, Virginia Polytechnic Institute, Blacksburg.
Thesis: "An Automated Personnel Identification System for the VPI Computing Center."

B. S. in Physics, 1976, Louisiana State University, Baton Rouge. Minor: Chemistry

Academic Experience

Assistant Professor, 1986-Present, Department of Computer Science, University of Pittsburgh, Pittsburgh, Pennsylvania.

Instructor/Research Associate, 1979-1986, Department of Computer Science, Louisiana State University, Baton Rouge, Louisiana.

Research Assistant, 1977-1979, Hardware and Network Services, Virginia Polytechnic Institute Computing Center, Virginia Polytechnic Institute, Blacksburg, Virginia.

Current and Completed Research Grants:

"Coincident Pulse Techniques for Hybrid Optical-Electronic Computer Systems" Air Force Office of Scientific Research, July 1989-July 1992, $479,511, (Co-PI with Rami Melhem and Steven Levitan).

"Parallel Memory Addressing Using Optical Pulse Delay Modulation" Air Force Office of Scientific Research, July 1988-July 1989, $50,132. (Co-PI with Rami Melhem and Steven Levitan).

"Optical Technology for Network Based Multiprocessors" NSF/MIPS, October 1988-October 1989, $49,983, (Co-PI with Rami Melhem and Steven Levitan).

"A VLSI Design and Test Facility for the University of Pittsburgh, NSF, CISE Research Instrumentation, January 1988, $65,597. (Co-PI with Steven P. Levitan).

"A High Performance Computer for Factoring Large Numbers," National Security Agency MDA904-85-H-0006, February 1985 February 1987, $198,296. (Research Associate with Walter G. Rudd).

"A High Performance Computer for Factoring Large Numbers" (equipment grant), National Science Foundation DCR 83-115-80, April 1985, $40,261 (Research Associate with Walter G. Rudd).

"Provision of Aquifer and Subsurface Database System", Louisiana Department of Natural Resources, Interagency Contract #21541-80-01, April 1980 - September 1980, $67,000 (Investigator with Walter G. Rudd).

Travel Grants

SIGDA, 28th Design Automation Conference, San Francisco, CA, June 1991.

Industrial and Internal Grants

"A Microcomputer Lab for the Department of Computer Science", (equipment grant) NCR Corporation, $100,000, December 1988.

"A Medium Speed Print Server for the Computer Science Department and the NCR Microcomputer Lab", University of Pittsburgh Provost's Departmental Infrastructure Grant, December 1990, $18,091.

"Terminal Server Replacement on the Computer Science Department LAN" University of Pittsburgh Provost's Departmental Infrastructure Grant, December 1990, $30,798.

Pending Grants

"Performance Driven, Multi-Level Synthesis Tools and Accompanying Design Environments", DARPA/ISTO/Pennsylvania State University, $1,900,757 (Co-PI with D.E. Setliff and S.P. Levitan, Department of EE, and M.J. Irwin, R.M. Owens and B. Pangrle, Penn State University).

"Novel Statistical and Computational Algorithms for Genetic Mapping", project within "A Human Genome Research Center for the Mapping of Chromosome 13", National Institutes of Health, five year project budget 1,136,485. (Co-PI with D. Weeks and H. Nicholaus).

Patents

Processor Utilizing Reconfigurable Process Segments, Co-inventors: W. G. Rudd, and D. A. Buell, Reg #4748585, May 1988.

An Optical Selector Switch, Co-inventors: R. Melhem, and S. Levitan, Reg #4883344, September 1988.

Refereed Publications

D. M. Chiarulli, R. Dilmore, S. Levitan, and R.G. Melhem, An All Optical Addressing Circuit: Experimental Results and Scalability Analysis (in press) *IEEE Journal of Lightwave Technology*.

C. Qiao, R.G. Melhem, D.M. Chiarulli, and S.P. Levitan, Multicasting in Optical Bus Connected Processors Using Coincident Pulse Techniques, (in press) *International Journal of Optical Computing*.

Z. Guo, R. Melhem, R. Hall, D. Chiarulli, S. P. Levitan. "Pipelined Communications in Optically Interconnected Arrays", *Journal of Parallel and Distributed Computing*, Vol. 12, No. 3, 1991.

D. M. Chiarulli, S. Levitan and R. Melhem, "Demonstration of an All Optical Addressing Ciruit" Technical Digest: OSA Topical Meeting on Optical Computing, 1991.

D. M. Chiarulli, R. Melhem, S. P. Levitan, "Asynchronous Control of Optical Buses for Distributed Multiprocessors" *Journal of Parallel and Distributed Computing* Vol. 10, No. 1, September 1990.

Z. Guo, R. Melhem, R. Hall, S. Levitan and D. Chiarulli, "Array Processors with Pipelined Optical Buses," *Proceedings* Frontiers 90 Conference on Massively Parallel Computation, October 1990.

D. M. Chiarulli, S. Levitan and R. Melhem, "Self Routing Interconnection Structures Using Coincident Pulse Techniques," *Proceedings* SPIE International Symposium on Advances in Interconnections and Packaging, November 1990.

Z. Guo, R. Melhem, R. Hall, S. Levitan and D. Chiarulli, "Pipelined Communications on Optical Busses", *Proceedings* SPIE International Symposium on Advances in Interconnections and Packaging, November 1990.

A. Martello, S.P. Levitan, D. Chiarulli, "Timing Verification Using HDTV", *Proceedings* Design Automation Conference, 1990.

S.P. Levitan, D. Chiarulli, R. Melhem, "Coincident Pulse Techniques for Multiprocessor Interconnection Structures", *Applied Optics* Vol. 29, May 1990.

R. Melhem, D. Chiarulli, S.P. Levitan, "Space multiplexing of optical waveguides in a distributed multiprocessor" *The Computer Journal, British Computer Society*, Vol. 3, No. 4, 1989.

D. M. Chiarulli, R. Melhem, S. P. Levitan, Parallel Memory Addressing Using Coincident Optical Pulses", *IEEE Computer*, Vol. 30, No. 12, December 1987.

M. Martin, D. Chiarulli, and S. Iyengar, "Parallel Processing of Quadtrees on a Horizontally Reconfigurable Architecture Computing System, *Proceedings*, 15th International Conference on Parallel Processing, Chicago, Illinois, August 1986, 895-902.

D. M. Chiarulli, W. G. Rudd, and D. A. Buell, "A Hierarchical Condition Code Structure for Parallel Architectures", SIAM Conference on Parallel Processing for Scientific Computing, Norfolk, Virginia, November 1985.

D. M. Chiarulli and D. A. Buell, "Parallel Microprogramming Tools for a Horizontally Reconfigurable Architecture," *International Journal of Parallel Programming*, Vol 15, No. 2, May 1987.

D. M. Chiarulli, W. G. Rudd, and D. A. Buell, "DRAFT--A Dynamically Reconfigurable Processor for Integer Arithmetic," *Proceedings*, 7th International Symposium on Computer Arithmetic, Urbana, Illinois, June 1985, 309-317.

W. G. Rudd, D. A. Buell, and D. M. Chiarulli, "A High Performance Factoring Machine," *Proceedings*, 11th Annual International Symposium on Computer Architecture, Ann Arbor, Michigan, June 1984, 297-300.

Invited Presentations

"Panel on the Future of Optical Computing," Supercomputing 91, Albuquerque NM, November 1991 (Invited Panelist).

"Temporal Parallelism in Optoelectronic Neural Networks," Jackson Hole Conference on Optical Neural Networks, February 1990.

"Coincident Pulse Techniques for Multiprocessor Interconnection," Optoelectronic Computing Systems Engineering Research Center, University of Colorado, February 1990.

"Optical Computing: When and How", Computer Science Department, Oregon State University, June 1987.

Published Reviews

Review of: Robert Spence, **Circuit Analysis by Computer: from Algorithms to Package,** Prentice/Hall, Engelwood Cliffs NJ, 1986, in *Computing Reviews,* Vol. 28, No. 2, April 1987.

Review of: James Archibald and Jean-Loup Baer, Cache Coherence Protocols: Evaluation Using a Multiprocessor Simulation Model, ACM Transations on Computer Systems, Vol. 4, No. 4, Nov 1986, in *Computing Reviews,* Vol. 28, No. 2, June 1987.

Review of: Forrest M. Mims, **Engineers Notebook,** McGraw Hill, New York NY, 1986, in *Computing Reviews,* Vol. 28, No. 2, June 1987.

Review of: Saul Ritterman, **Computer Circuit Concepts,** McGraw Hill, New York NY, 1986, in *Computing Reviews,* Vol. 28, No. 2, August 1987.

Review of: Marco Annaratone, **Digital CMOS Circuit Design,** Kluwer Academic Publishers, Boston MA, 1986, in *Computing Reviews,* Vol. 28, No. 2, September 1987.

Review of: G. J. Lipovski, **Single and Multiple-chip Microcomputer Interfacing,** Prentice/Hall, Engelwood Cliffs NJ, 1988, in *Computing Reviews,* Vol. 30, September 1989.

Technical Reports

D. M. Chiarulli, "A Fast Multiplier for Very Large Operands," University of Pittsburgh, Computer Science Technical Report 86-011

D. M. Chiarulli, "A VLSI Partitioning of the Draft Architecture", LSU Computer Science Technical Report 85-049.

Courses Taught

Graduate and Undergraduate Computer Architecture

Graduate and Undergraduate Operating Systems

Graduate Seminar in Optical Computing

Graduate Seminar in VLSI Design

Systems Programming

Assembly Language Programming

Students Supervised

Greg Owens, M.S. 1988, "A Simulation of Data Collections Topologies in an Optically-Addressed Parallel Memory."

Kausalai Wijekumar, M.S. 1989, "SPLICE: Connecting Synthesis and Verification for VLSI Design."

Brian Field, M.S. 1989, "An Extension of *Magic* to Support Remote VLSI Design Verification."

Mike Durbin, M.S. 1990, "Likelihood, A Computer Program for Genetic Linkage Analysis."

Patricia Rapp, M.S. 1991, "Effect of the Design Hierarchy on Memory Requirements for Incremental Simulation."

Robert Ditmore, M.S. 1991, "An Analyis of Power Distribution in Optical Busses."

James Plusquellic, (Ph.D. in progress), "VLSI Component Testing by Analysis of Switching Transients."

Student Committees and Examining Boards

Karen Meyer-Arendt, M.S. (CS, Oregon State Univ.), 1987, "Parallel Code Generation via Scheduling."

Robert Swarner, M.S. (EE), 1988, "Applying Mandatory Security for UNIX to Operating in a Multi-Level Environment."

John Elliot, B.S. (University Honors College), 1988, "SCED: An Icon Based Schematics Editor."

Ron Vali, Ph.D. (EE), 1989, "Memory Architectures for Supercomputers."

Melanie D. Berg, M. S. (EE), 1991, "Error Bounding Parallel Simulated Annealing on a Hypercube."

Professional Societies and Service

Member of IEEE Computer Society
Member of Optical Society of America
Member of International Society for Optical Engineering (SPIE)
Evaluation Board member for Ben Franklin Foundation Challenge Grants
Reviewer for National Science Foundation
Textbook review for Boyd and Frasier Publishing
Textbook review for Prentice Hall Publishing
Referee for Applied Optics
Referee for IEEE Computer
Referee for IEEE Tranactions on Computers
Referee for Journal of Parallel and Distributed Computing
Referee for International Symposium on Computer Arithmetic
Referee for International Conference on Parallel Processing

University Service

Executive Committee for Academic Computing (ECAC), 1989-1991, This is the primary faculty oversight committee for Computing and Information Services.

ECAC Mainframe Services Subcommittee, 1988-1989, Formerly the Performance and Service Priorities Subcommittee, responsibilities include the oversight of computing center services relating to the VAX cluster and VAX ULTRIX systems. Specific contributions include a set of performance measures which were adopted for monthly reporting to the committee.

ECAC Budget and Planning Subcommittee, 1989-1991, This subcommittee is responsible for review of all ECAC budgetary matters. The committee has also periodically produced long range planning documents on university wide computing.

EE/CS Computer Engineering Committee, 1987-1991, This committee is evaluating curriculum and administrative issues related to introduction of a Computer Engineering program.

Departmental Service

Computing Needs Committee (Chairman), 1987-1991, This committee evaluates departmental computing resources on an annual basis to establish aquisition priorities.

Graduate Program Committee (Chairman), 1990-1991, This committee is responsible for curriculum, new courses, and administrative matters relating to graduate studies as well as the annual administration of the preliminary exam. A specific contribution while chairman was a significant reorganization of the rules for the preliminary exam.

Undergraduate Curriculum Committee, 1989-1990, As a member of this committee, I participated in the restructuring of of the undergraduate core courses for computer science, including a conversion to lecture/recitation formats. Specific contributions include an overhaul of the undergraduate systems course sequence.

Curriculum Vitae

Steven Peter Levitan

Department of Electrical Engineering

Benedum Engineering Hall

University of Pittsburgh

Pittsburgh, PA 15261

PROFESSIONAL INTERESTS

The design, modeling, and simulation of highly parallel systems, including parallel computer architectures, parallel algorithms, and VLSI. Additional interests include design tools and methodology for software, hardware, and VLSI.

CURRENT POSITION

Wellington C. Carl Assistant Professor in the Department of Electrical Engineering at the University of Pittsburgh.

EDUCATION

Ph. D. May 1984 University of Massachusetts Department of Computer and Information Science (COINS). Dissertation title: "Parallel Architectures and Algorithms: A Programmer's Perspective". Advisor: Caxton C. Foster.

M. S. September 1979 University of Massachusetts, (COINS), Specialization: Computer Systems.

B. S. June 1972 Case Western Reserve University, School of Engineering. Major: Computer Science, Minor: Electrical Engineering

PROFESSIONAL POSITIONS HELD

Wellington C. Carl Assistant Professor, 1987-: Department of Electrical Engineering, University of Pittsburgh.

Assistant Professor, 1985-1986 (tenure stream began Sept., 1985): Department of Electrical and Computer Engineering (ECE), University of Massachusetts, Amherst. **Director, VLSI Laboratory**(1985-1986): Responsible for coordination and direction of VLSI design laboratory. Liaison to the Massachusetts Technology Park Corporation, Massachusetts Microelectronics Center (MMC).

Visiting Assistant Professor, 1984-1985: Electrical and Computer Engineering, University of Massachusetts, Amherst.

Consultant, 1984-1987: Viewlogic Systems Inc. Developed VLSI design and simulation software.

Consultant, 1982-1983: Digital Equipment Corporation. Consulted for the VLSI Advanced Architectures Group on silicon compilers, simulation, and parallel processing issues.

Engineer, Summer 1982: DEC. One of the team which developed the "Silicon Synthesis Project", a VLSI design tool.

Co-founder, 1980-1983: Humanistic Computing Systems, consultants in the development of user-friendly software.

Teaching Assistant/Lecturer, 1978-1982: Department of Computer and Information Science (COINS), University of Massachusetts, Amherst.

Senior Systems Engineer, 1972-1977: Xylogic Systems Inc. Designed serial, parallel and DMA interfaces for minicomputer based text processing systems used in newspaper production. I trained and supervised in house test and field service personnel. Largest project was a multi-computer, dual-chain disk controller.

Test Technician, 1972: ARP Inc. Tested and repaired music synthesizers, and trained repair personnel.

PUBLICATIONS

Refereed Journal Publications:

1. "SPAR: A Schematic Place and Route System"; Stephen T. Frezza and Steven P. Levitan; (submitted) *IEEE Transactions on Computer Aided Design of Integrated Circuits.*

2. "A Systems Theoretic Approach to the Functional Characterization of the Hippocampal Formation"; R.J. Sclabassi, D.N. Krieger, German Barrionuevo, S.P. Levitan, T.W. Berger; (submitted) *Annals of Biomedical Engineering*, 1991.

3. "Optical Multicasting in Linear Arrays"; Chunming Qiao, R.G. Melhem, S.P. Levitan and D.M. Chiarulli,(in press) *International Journal on Optical Computing.*

4. "An All Optical Addressing Circuit: Experimental Results and Scalability Analysis"; Donald M. Chiarulli, Robert M. Ditmore, Steven P. Levitan, and Rami G. Melhem;*IEEE Journal of Lightwave Technology, Vol. 9,* No. 12, pp. 1717-1725, 1991.

5. "Pipelined Communications In Optically Interconnected Arrays"; Z. Guo, R.G. Melhem, R.W. Hall, D.M. Chiarulli, and S.P. Levitan; *Journal of Parallel and Distributed Computing, Vol. 12,* No. 3, pp. 269-282, 1991.

6. "An Interactive Toolset for Characterizing Complex Neural Systems"; D.N. Krieger, T.W. Berger, S.P. Levitan, and R.J. Sclabassi; *Computers and Mathematics, Vol. 20,* Mathematical Models in Medicine, No.4-6, pp. 231-246, 1990.

7. "Coincident Pulse Techniques for Multiprocessor Interconnection Structures"; S.P. Levitan, D.M. Chiarulli, R.G. Melhem; *Applied Optics; Vol. 29,* No. 14, pp. 2024-2033, May, 1990.

8. "Optical Bus Control for Distributed Multiprocessors"; D.M. Chiarulli, S.P. Levitan, R.G. Melhem; *Journal of Parallel and Distributed Computing; Vol. 10,* No. 1, pp. 45-54, 1990.

9. "Space Multiplexing of Optical Waveguides in a Distributed Multiprocessor"; R.G. Melhem, D.M. Chiarulli and S.P. Levitan; *The Computer Journal, British Computer Society, Vol. 32,* No. 4, pp. 362-369, 1989.

10. "The Image Understanding Architecture"; C. C. Weems, S. P. Levitan, A. R. Hanson, E. M. Riseman, J. G. Nash, D. B. Shu; *International Journal of Computer Vision Vol. 2,* pp. 251-282 (1989).

11. "Using Coincident Optical Pulses for Parallel Memory Addressing"; D. Chiarulli, R. Melhem, S. Levitan; *Computer* Vol. 20, No. 12, pp. 48-57. December, 1987.

Chapters in Edited Books:

1. "Nonlinear Systems Analysis of Network Properties of the Hippocampal Formation"; T.W. Berger, G. Barrionuevo, S.P. Levitan, D.N. Krieger, and R.J. Sclabassi; pp. 283-352; *Learning and Computational Neuroscience: Foundations of Adaptive Networks,* M. Gabriel and J. W. Moore (Eds.), M.I.T. Press, 1990.

2. "Theoretical Decomposition of Neuronal Networks"; R.J. Sclabassi, D.N. Krieger, J. Solomon, J. Samosky, S.P. Levitan, and T.W. Berger; (in) *Advanced Methods of Physiological System Modeling, Vol. 2,* V.Z. Marmarelis (Ed.), pp. 129-146, Plenum Press, New York, 1989.

3. "Using VHDL as a Language for Synthesis of CMOS VLSI Circuits"; S.P. Levitan, A.R. Martello, R.M. Owens, M.J. Irwin; (in) *Computer Hardware Description Languages and their Applications* J.A. Darringer and F. J. Ramming, Eds.; Elsevier, Amsterdam, 1989; pp. 331-346; IFIP WG 10.2, 9th Intl. Symp. on Computer Hardware Description Languages; Washington D.C., June, 1989.

4. "The UMass Image Understanding Architecture"; Steven P. Levitan, Charles C. Weems, Allen R. Hanson and Edward M. Riseman; (in) *Parallel Computer Vision;* Leonard Uhr (Ed.), Academic Press, New York, 1987; pp. 215-248.

5. "Measuring Communication Structures in Parallel Architectures and Algorithms"; Steven P. Levitan (in) *The Characteristics of Parallel Algorithms;* L. Jamieson, D. Gannon, and R. Douglass (Eds.), Cambridge, MA; MIT Press, 1987; pp. 101-137.

6. "Signal to Symbols: Unblocking the Vision Communications/Control Bottleneck"; Steven P. Levitan, Charles C. Weems, Edward M. Riseman; (in) *VLSI Signal Processing* (proceedings of the 1984 IEEE Workshop on VLSI Signal Processing at University of Southern California, Los Angeles, CA; November 27-29, 1984); IEEE Press; New York, NY; 1984; pp. 411-420.

7. "A Content Addressable Array Parallel Processor and Some Applications"; Charles C. Weems, Steven P. Levitan, Daryl T. Lawton, and Caxton C. Foster; (in) *Image Understanding,* Proceedings of the DARPA Workshop, Arlington, Virginia; June 23, 1983; Science Applications, Inc. Report Number SAI-84-176-WA.

Refereed Conference Proceedings:

1. "Efficient Channel Allocation for Routing in Optically Interconnected Mulitprocessor Systems"; C. Qiao, R. Melhem, D. M. Chiarulli, S. P. Levitan; **SPIE Symposium on OE/Aerospace Sensing'92, Conf. on Advances in Optical Information Procesing V**; Orlando, Fl.; 1704-25; April 20-24, 1992.

2. "Temporal Specification Verification via Causal Reasoning"; A. R. Martello, S. P. Levitan; **Tau'92: ACM International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems**; Princeton, New Jersey; March 18-20, 1992.

3. "Architectural Synthesis via VHDL"; S.P. Levitan, B. Pangrle, Y.W.Hsieh; **Third Physical Design Workshop**; Nemacolin Woodlands, PA, May 20-23, 1991.

4. "Multicasting in optical bus connected processors using coincident pulse techniques"; Chunming Qiao, R. Melhem, D. Chiarulli and S. Levitan; **International Conference on Parallel Processing**; (poster); St. Charles, IL, August 20-23, 1991.

5. "Demonstration of an All Optical Addressing Circuit"; D. Chiarulli, S. Levitan, R. Melhem; **Optical Society of America Topical Meeting on Optical Computing**; Technical Digest Vol. 6, TuC3-1, pp.235-238; Salt Lake City, UT, March 4-6, 1991.

6. "Self Routing Interconnection Structures Using Coincident Pulse Techniques"; D. Chiarulli, S. Levitan, R. Melhem; **SPIE OE/Boston'90**, 1390-25, S4, pp. 403-414; November 4-9, 1990.

7. "Pipelined Communications on Optical Busses"; Z. Guo, R. Melhem, R. Hall, D. Chiarulli, S. Levitan; **SPIE OE/Boston'90**, 1390-26, S4, pp. 415-426; November 4-9, 1990.

8. "The Identification of Hippocampal Network Function"; R.J. Sclabassi, D.N. Krieger, G.Barrionuevo, S.P. Levitan, and T.W. Berger; **Proceedings of the IEEE Annual International Conference of Engineering in Medicine and Biology Society**; Vol. 12, No. 4, pp.1886-1888; Philadelphia, October, 1990.

9. "A Fault Tolerant Design of the Generalized Cube Network"; T.D Han, D.A. Carlson, and S.P Levitan; **Proceedings of the ISMM International Conference on Parallel and Distributed Computing, and Systems**; pp. 160-165; October 10-12, New York; R.A. Ammar, Editor; Acta Press, 1990.

10. "Array Processors with Pipelined Optical Busses"; Z. Guo, R. Melhem, R. Hall, D Chiarulli, S. Levitan; **IEEE Frontiers'90: 3rd Symposium on the Frontiers of Massively Parallel Computation**; pp. 333-342; University of Maryland, College Park, MD, October 8-10, 1990.

11. "Causal Timing Verification"; A. R. Martello, S. P. Levitan; **Tau'90: ACM International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems**; Vancouver, BC; August 15-17, 1990.

12. "Timing Verification Using HDTV"; A. R. Martello, S. P. Levitan, D. M. Chiarulli; **Proceedings of the 27th Design Automation Conference**, pp. 118-123; June 1990.

13. "Modeling of Neuronal Networks Through Decomposition"; R.J. Sclabassi, J. Samosky, D.N. Krieger, J. Solomon, S.P. Levitan, and T.W. Berger; **International Joint Conference on Neural Networks**; pp. I773-I780; Washington, D.C. June 1989.

14. "A VLSI CAD System for VHDL"; S. P. Levitan, R. M. Owens, M. J. Irwin; **Colorado Microelectronics Conference**; pp. 1-8; Antlers Hotel, Colorado Springs, CO March 30-31, 1989.

15. "An Input/Output Model of the Hippocampal Formation"; Robert J. Sclabassi, Don Krieger, Jacqueline Solomon, Steven P. Levitan, German Barrionuevo, Theodore Berger; **Society for Neuroscience Abstracts**; 14th Annual Meeting of the Society for Neuroscience; Vol. 14, p. 247; Toronto, November 13-18th 1988.

16. "An External Network Model of the Hippocampal Formation"; Robert J. Sclabassi, Don Krieger, Jacqueline Solomon, Steven P. Levitan, German Barrionuevo, Theodore Berger; **International Neural Network Society Abstracts**, Neural Networks (Supplement) Conference Proceedings; Boston, MA; September, 1988; vol. 1, p. 273.

17. "A Neurophysiologic Neural Network Model"; Don Krieger, Jacqueline Solomon, Steven Levitan, Theodore Berger, German Barrionuevo, Robert Sclabassi; **19th Annual Pittsburgh Conference on Modeling and Simulation**; May 5-6, 1988; vol. 19, pp. 2397-2401.

18. "An Easily Reconfigurable, Circuit Switched Connection Network"; Deepak Rana, Charles C. Weems, and Steven P. Levitan; **IEEE International Symposium on Circuits and Systems**; Helsinki University of Technology; Espoo, Finland; June 7-9, 1988.

19. "Teaching Computer Architecture as Engineering Design with VLSI"; S. P. Levitan and J. T. Cain; **21st Annual Hawaiian International Conference on Systems Sciences** (*HICSS*); pp. 85-90; Kona, HI, January 5-8, 1988.

20. "VLSI Design of High-Speed, Low-Area Addition Circuitry"; Tack-Don Han, David A. Carlson and Steven P. Levitan; **IEEE Intl. Conference on Computer Design** (ICCD); pp. 418-422; Port Chester, NY, October 5-8, 1987.

21. "The Image Understanding Architecture"; Charles C. Weems, Steven P. Levitan, Allen R. Hanson and Edward M. Riseman; **Proceedings of the DARPA Image Understanding Workshop**; pp. 483-496; Los Angeles, CA; February, 1987;

22. "A Testable, Asynchronous Systolic Array Implementation of an IIR Filter"; Deepak Rana, Steven P. Levitan, David A. Carlson and Charles E. Hutchinson; **Custom Integrated Circuits Conf.**; pp. 90-93; Rochester, NY, May 12-15 1986.

23. "Evaluation Criteria for Communication Structures in Parallel Architectures"; Steven P. Levitan; **1985 International Conference on Parallel Processing**; St. Charles, Ill. August 20-23, 1985.

24. "Iconic to Symbolic Processing Using a Content Addressable Array Parallel Processor"; D. Lawton, S. Levitan, C. Weems, E. Riseman, A. Hanson, M. Callahan; **Proceedings, SPIE Int. Soc. Opt. Engr.**, Vol. 504 pp. 92-111 (1984) (Applications of Digital Image Processing VII, San Diego, CA. August 12-24, 1984).

25. "Iconic and Symbolic Processing Using a Content Addressable Array Parallel Processor"; C. Weems, D. Lawton, S. Levitan, E. Riseman, A. Hanson; **Proceedings of the IEEE Computer Society Conf. on Computer Vision and Pattern Recognition**; pp. 598-607; San Francisco, CA; June 19-29, 1985.

26. "Parallel Processing of Iconic to Symbolic Transformation of Images"; D. I. Moldovan, C. I. Wu, J. G. Nash, S. P. Levitan, C. Weems; **Proceedings of the IEEE Computer Society Conf. on Computer Vision and Pattern Recognition**; pp. 257-264; San Francisco, CA, June 19-29, 1985.

27. "Iconic to Symbolic Processing Using the Content Addressable Array Parallel Processor"; Daryl T. Lawton, Steven P. Levitan, Charles C. Weems, Edward M. Riseman, and Allen R. Hanson; **Proceedings of the 1984 Fall Image Understanding Workshop**; New Orleans, LA, October, 1984.

28. "Development and Construction of a Content Addressable Array Parallel Processor for Knowledge-Based Image Interpretation"; Charles C. Weems, Steven P. Levitan, Caxton C. Foster, Edward M. Riseman, Daryl T. Lawton, and Allen R. Hanson; **Workshop on Algorithm-Guided Parallel Architectures for Automatic Target Recognition**; Leesburg, VA; July 16-18, 1984.

29. "Titanic: A VLSI Based Content Addressable Parallel Array Processor"; Charles C. Weems, Steven P. Levitan, and Caxton C. Foster; **International Conference on Computer Circuits**, New York, NY, September 29 - October 1, 1982.

30. "Algorithms for a Broadcast Protocol Multiprocessor"; Steven P. Levitan, **3rd International Conference on Distributed Computing Systems**, Miami/Ft. Lauderdale, FL, October 18-22, 1982.

31. "Finding an Extremum in a Network"; Steven P. Levitan and Caxton C. Foster, **9th Annual International Symposium on Computer Architecture**, Austin, TX, April 26-29, 1982.

32. "Real-Time LISP Using Content Addressable Memory"; Jeffrey G. Bonar and Steven P. Levitan; **1981 International Conference on Parallel Processing**, Bellaire, MI, August 25-28, 1981.

Technical Reports and Popular Journals:

1. "Fifth Semi-Annual Keystone Research Group Meeting May 3, 1991" M. J. Irwin, R. M. Owens, B. M. Pangrle, S. P. Levitan, D. M. Chiarulli, and D. E. Setliff; Department of Computer Science, The Pennsylvania State University, CS-91-13 June, 1991.

2. "A VHDL Design Environment"; A.R. Martello and S.P. Levitan; SIGDA Newsletter, Vol. 20, No. 3, pp. 52-67; December, 1990.

3. "Fourth Semi-Annual Keystone Research Group Meeting November 19, 1990" S. P. Levitan, D. M. Chiarulli, D. E. Setliff, M. J. Irwin, R. M. Owens, and B. M. Pangrle; Department of Electrical Engineering, University of Pittsburgh TR-CE-90-002

4. "The Keystone Design Environment: Philosophy and Methodology"; S.P. Levitan, D.E. Setliff, D.M. Chiarulli, M.J. Irwin, R.M. Owens, and B.Pangrle; TR-CE-91-001, Electrical Engineering, University of Pittsburgh, November, 1990.

5. "Selected Topics in Architecture, Logic and Physical Synthesis"; M.J. Irwin, R.M. Owens, and S.P. Levitan, TR-CE-88-004, Electrical Engineering, University of Pittsburgh, November, 1988.

6. "Sced: An Icon Based Schematic Editor"; John P. Elliott and Steven P. Levitan; PICA Laboratory Technical Report TR-CE-88-001, Electrical Engineering, University of Pittsburgh, July, 1988.

7. "Asynchronous Control of Optical Busses in Closely Coupled Distributed Systems"; Donald M. Chiarulli, Rami Melhem, Steven P. Levitan; Technical Report 88-2, Department of Computer Science, University of Pittsburgh, 1988.

8. "The Image Understanding Architecture"; C. C. Weems, S. P. Levitan, A. R. Hanson, E. M. Riseman, J. G. Nash, D. B. Shu; COINS Technical Report 87-76; University of Massachusetts at Amherst; 1987.

9. *Parallel Algorithms and Architectures: A Programmer's Perspective*, Steven P. Levitan; COINS Technical Report 84-11; University of Massachusetts at Amherst; May 1984.

10. "APP-L-ISP" (product review); Jeffrey G. Bonar and Steven P. Levitan, *BYTE* June 1982.

11. "Three Microcomputer LISPs"(product review); Steven P. Levitan and Jeffrey G. Bonar, *BYTE*, August 1981

12. "The Super-Kim Project: A 6502 Microcomputer System for the Real-Time Laboratory"; Steven P. Levitan, COINS Technical Report. July 1979.

13. "CAMEOS: a Content Addressable Memory Enhanced Operating System"; Steven P. Levitan and Caxton C. Foster, COINS Technical Report. March 1978.

Invited Presentations and Workshops:

1. "Optical MIMD Architectures"; **AFOSR Workshop on Reconfigurable Optical Interconnects**; Boulder, CO, March, 1992.

2. "Panel on the Future of Optics in Computing"; **Supercomputing '91,** Albuquerque, NM, November, 1991 (Panel Chair).

3. "Keystone: A VHDL Simulation and Synthesis Environment for VLSI Design"; **IBM Thomas J. Watson Research Center** Hawthorne, NY, October, 1991.

4. "Optical Interconnection Structures For Multiprocessor Applications"; **University of Pittsburgh,** Department of Electrical Engineering, January, 1991.

5. "Using Keystone for Verification and Synthesis"; **Viewlogic Systems**, Marlboro, MA, September, 1990.

6. "Timing Verification of Digital Interfaces"; **Carnegie Mellon University,** Department of Electrical and Computer Engineering, May, 1990.

7. "Optical Parallel Processing"; **Workshop on Optical Neural Networks,** Jackson Hole, WY, February, 1990.

8. "Addressing and Control in Optical Interconnection Networks for Hybrid Multiprocessors"; **University of Colorado at Boulder,** Optoelectronic Computing Systems Center, Boulder, CO. February, 1990.

9. "The Keystone Silicon Synthesis Project"; **Viewlogic Systems,** Marlboro, MA January, 1990.

10. "Silicon Synthesis: A VHDL Approach"; IEEE Student Chapter, **Pennsylvania State University.** November, 1989.

11. "Experiences Using VHDL in the Classroom"; **VHDL Users Group Meeting,** Sheraton Hotel, Redondo Beach, CA,October, 1989.

12. "Synthesis of CMOS Structures from VHDL"; **VHDL Methods Workshop,** University of Virginia, Charlottesville, VA; August, 1989.

13. "VLSI Curriculum: CAD for VLSI"; **VLSI Education Conference & Exposition,** Santa Clara, CA, July, 1989, pp. 181-182.

14. "From VHDL to Layout"; **VHDL Users Group Meeting,** Sheraton Hotel, Redondo Beach, CA, October, 1988.

15. "An Integrated Capture and Simulation Tool for Digital Designs"; **Penn State University,** September, 1988.

16. "Architectures and VLSI"; **Workshop on Special Computer Architectures for Robotics,** International Conference on Robotics and Automation, Philadelphia, April, 1988.

17. **NSF/MOSIS Undergraduate Education Workshop,** NSF, November, 1987.

18. "Parallel Algorithms and Architectures: A Programmers Perspective"; **Taxonomy of Parallel Algorithms Workshop,** Los Alamos National Laboratory, Santa Fe, New Mexico, November, 1983.

19. "Topics In Computer Architecture"; **Smith College,** February, 1983.

Patents

An Optical Selector Switch, (with R. Melhem, and D. Chiarulli), Approved September 1988, Number 4,883,334.

GRANTS

Current:

National Science Foundation, 5/92-5/95, "A Research Experiences for Undergraduates Site: Training Students to Model Polymer Behavior Through Computer Simulations"; $150,000 (CI) with A.C. Balazs (PI), and R.L. Pinkus.

National Science Foundation, 1/92-12/94, "Temporal Specification Verification"; $218,292 (PI).

GUIdance Technologies, 1/92-1/93, "Unrestricted Gift"; $17,929.

Association for Computing Machinery - SIGDA, 12/92-6/93, ACM/SIGDA "Creation of a SIGDA Internet Server"; $23,834 (PI).

National Science Foundation, 7/91-7/93, "Distribution of VLSI Design Software for Education and Research"; $97,618 (PI) MIP-9101656.

Association for Computing Machinery - SIGDA, 11/90-6/93, ACM/SIGDA "Equipment Support for Design Automation University Booth 1991-1993"; $20,000 (PI).

National Institute of Mental Health (ADAMHA), 4/90-3/93 "Contribution of PCP and NMDA Receptors to Network Properties of the Hippocampal Formation"; $600,000 ($44,915 to Electrical Engineering to date) (CI) with T. W. Berger(PI), R. J. Sclabassi(CI), G. Barrionuevo, D. N. Krieger. Program 1, part of $2,270,700 Behavioral Neuroscience and Schizophrenia grant under Edward M. Stricker (PI); MH45156-01A1.

Air Force Office of Scientific Research, 7/89-7/92, "Coincident Pulse Techniques for Hybrid Electronic/Optical Computer Systems"; $479,511 ($108,562 to Electrical Engineering to date) (CO-PI) with D.M. Chiarulli, R. Melhem; AFOSR-89-0469.

National Science Foundation, 10/87-6/89 "Application to Use DARPA/NSF Service (MOSIS) for Fabrication of Prototype Quantities of Custom Integrated Circuits to Support Education"; $15,200. Renewed 6/89-9/90 $14,900, Renewed 9/90-9/91 $5,940, Additional funding 1/91 $6,000, Renewed 10/91-9/92 $6,525 (PI).

Office of Naval Research, 6/87-5/90 "Changes in Neuronal Network Properties Induced by Learning and Synaptic Plasticity: A Nonlinear Systems Approach"; $394,591 ((CI) with T.W. Berger(PI), R.J. Sclabassi, G. Barrionuevo, D.N. Krieger). Supplement: 6/90-5/91 $137,352; ($8,741 to Electrical Engineering to date) N00014-87-K-0472.

Completed:

Air Force Office of Scientific Research, 10/88-9/91, "A System Theoretic Investigation of Neuronal Network Properties of the Hippocampal Formation"; $476,681 ($46,764 to Electrical Engineering to date) (CI) with T.W. Berger(PI), R.J. Sclabassi, G. Barrionuevo, D.N. Krieger; AFOSR-890197.

National Science Foundation, 5/91-7//91, "A Research Experiences for Undergraduates Site: Training Students to Use Computer Simulations as Research Tools"; $42,000 (CI) with A.C. Balazs (PI), and J.F. Patzer.

Viewlogic Systems, Inc., 6/91, "Software grant: Workview 750 system with 7400 simulation models",$13,000 (value) (PI).

The Ben Franklin Technology Center of Western Pennsylvania, 1/91-9/91, "Unix Graphic User Interface Development System"; $117,731 ($31,632 to Electrical Engineering) (CO-PI) with J. G. Bonar, GUIdance Technologies.

National Science Foundation, 5/88-11/90, "Instrumentation and Laboratory Improvement: Real Time Signal Processing Laboratory for Undergraduate Instruction"; $68,435 (CI) with L.F. Chaparro (PI), E.W. Kamen, S. Park; ENG-8852496.

National Science Foundation, 5/89-4/90, "Optical Technology for Network Based Multiprocessors"; $49,983 (CO-PI) with D. Chiarulli, R. Melhem; MIP-8901053.

Air Force Office of Scientific Research, 7/88-7/89, "Parallel Memory Using Coincident Optical Pulses"; $50,132 (CO-PI) with D. Chiarulli, R. Melhem; AFOSR-88-0198.

National Science Foundation, 1/88-1/89, "CISE Instrumentation: A VLSI Design and Test Facility for the University of Pittsburgh"; $65,597 (CO-PI) with D. Chiarulli; CCR-8716980.

DARPA/University of Massachusetts, Subcontract. 6/87-10/87, "Array Control Unit for the UMass Image Understanding Architecture"; $28,069; 10/87-12/88 additional funds $14,715; 8/88-12/88 additional funds $8,922 (PI).

Central Research Development Fund, University of Pittsburgh, 7/87-7/88, "An Integrated Tool Set for Digital Systems Design"; $9,900 (PI).

Advanced Research Projects Agency/Army, 9/86-12/88, "Image Understanding Architecture"; $1,752,200 (CI) with A. Hanson, E. Riseman, C. Weems); DACA76-86-C0015.

Advanced Research Projects Agency/AFOSR, 2/86-2/88, "Intermediate Level Computer Vision Processing Algorithm Development For Content Addressable Array Parallel Processor"; $197,000; (CI) with A. Hanson, E. Riseman, C. Weems; F49620-86-C-0041.

Naval Research Laboratory, 3/85-10/85, "Parallel Algorithms for Low, Intermediate, and High Level Image Understanding Tasks Using the Content Addressable Array Parallel Processor (CAAPP)"; $24,500; (CI) with A. Hanson, E. Riseman, C. Weems; N00014-85-K-2008.

National Science Foundation, 6/85-6/86 "Computer Research Equipment (Infrastructure)"; $105,413; (CI) with D. Carlson (PI), W. Kohler, M. Krishna, D. Pradhan, A. Singh, J. Stankovic, D. Towsley); DCR-8505499.

Pending:

DARPA/ISTO/Pennsylvania State University, 6/90-5/93 "Performance Driven, Multi-Level Synthesis Tools and Accompanying Design Environments"; $1,900,757; (CO-PI) with D.E. Setliff, and D.M. Chiarulli, Department of Computer Science and M.J. Irwin, R.M. Owens and B. Pangrle, Penn State University.

Vita of Rami G. Melhem
Department of Computer Science
The University of Pittsburgh
Pittsburgh, PA 15260.
(412)-624-8426, melhem@cs.pitt.edu

GENERAL INFORMATION:

Birthdate:	June 30, 1954
Birthplace:	Cairo, Egypt
Marital Status:	Married, two children
Languages:	Arabic and French

EDUCATION:

1983	Ph.D.	Computer Science, University of Pittsburgh
1981	M.S.	Computer Science, University of Pittsburgh
1981	M.A.	Mathematics, University of Pittsburgh
1978	B.S.	Mathematics, Ein-Shams University, Cairo, Egypt
1976	B.S.	Electrical Engineering, Cairo University, Egypt

PROFESSIONAL ACTIVITIES:

Member in	the IEEE Computer Society
	the Association for Computer Machinery
	The International Society for Optical Engineering - SPIE.
Referee for	IEEE Computer Magazine - IEEE Trans. on Computers
	IEEE Trans. on Automatic Control - SIAM Journal on Computing
	IEEE Trans. on Parallel and Distributed Computing - Distributed Computing
	Parallel Computing - J. of Parallel and Distributed Computing
	Journal of Computer and System Sciences - Computers and Structures
	The International Journal of Parallel Programming
	The International J. of Computer Simulation - J. of VLSI Signal Processing
	The International Journal of Supercomputer Applications
	Numerous conferences and symposia
Reviewer for	The National Science Foundation
Organizer	Symposium on PCGG methods and Supercomputing - Pittsburgh, PA - 1989.
Member	Program committee - Int. Conf. on Application Specific Array Processors - 1991.
	Program committee - ISMM Conf. on Parallel and Distributed Comp. & Sys. - 1991.
	Program committee - Int. Workshop on Defect and Fault Tolerance in VLSI - 1992.
Chairman	Program committee - ISMM Conf. on Parallel and Distributed Comp. & Sys. - 1992.
Guest editor	J. of Parallel and Distributed Computing- Sepecial issue on Optical Computing - 1993.

PROFESSIONAL EXPERIENCE:

1984	Research Associate, University of Pittsburgh (January-September)
1984-1987	Assistant Professor of Computer Science, Purdue University (on leave from Sept. 1985 to Sept. 1987)
1985-1986	Visiting Assistant Professor of Mathematics, University of Pittsburgh (September-August)
1986-1989	Assistant Professor of Computer Science, University of Pittsburgh (September-August)
1989-	Associate Professor of Computer Science, University of Pittsburgh (September-)

GRANT AWARDS:

ONR: "Application of Computational Networks and Systolic Arrays to Scientific Computation". With W. C. Rheinboldt. Total Award: $233,402.00. June 1985 to September 1988.

AFOSR: Investigator, (C. Hall and T. Porsching, principal Investigators); "Computational Fluid Dynamics at the Institute for Comp. Math. & Applications" Total Award: $587,858.00, June 1984 to June 1987.

AFOSR: "Parallel Memory Addressing Using Coincident Optical Pulses". With D. Chiarulli and S. Levitan. Total Award: $50,132.00. July 1988 to July 1989.

NSF: "Optical Technology in Network Based Multiprocessors". With D. Chiarulli and S. Levitan. Total Award: $49,983.00. July 1989 to June 1990.

AFOSR: "Coincident Pulse Techniques for Hybrid Optical-Electronic Computer Systems". With D. Chiarulli and S. Levitan. Total Award: $479,511.00. August 1989 to July 1992.

NSF: "Bi-level Reconfigurations of Fault Tolerant Arrays in Bi-modal Environments". Total Award: $61,547.00. September 1989 to August 1991.

NSF: "CISE Research Instrumentation grant for the acquisition of an Intel Hypercube". With M.L. Soffa and T. Znati. Total Award: $124,300.00. March 1990 to March 1991.

CURRENT RESEARCH INTERESTS:

Parallel and distributed computing - Fault tolerance in large computational networks
Optical Computing - Special Purpose Architectures - Scientific Computing

PATENTS:

"An Optical Selector Switch", Co-inventors: D. Chiarulli and S. Levitan.
Patent number 4,883,334 - November 28, 1989.

PUBLICATIONS IN ARCHIVED JOURNALS

1) R. G. Melhem and W. C. Rheinboldt, "A Comparison of Methods for Determining Turning Points of Non-linear Equations", Computing, vol. 29, pp.201-226, (1982).

2) R. G. Melhem and W. C. Rheinboldt, "A Mathematical Model for the Verification of Systolic Networks", SIAM Journal on Computing, vol. 13, no. 3, pp. 541-565, (1984).

3) R. G. Melhem, "On the Design of a Pipelined/Systolic Finite Element System", Computers and Structures, vol. 20, pp.67-75, (1985).

4) R. G. Melhem, "Formal Analysis of a Systolic System for Finite Element Stiffness Matrices", Journal of Computer and System Sciences, vol. 31, no. 1, pp. 1-27, (1985).

5) R. G. Melhem, "A Study of Data Interlock in Computational Networks for Sparse Matrix Multiplication", IEEE Transactions on Computers, vol 36, no 9, pp.1101-1107, (1987).

6) R. G. Melhem, "Parallel Gauss/Jordan Elimination for the Solution of Dense Linear Systems". Parallel Computing, vol 4, no 3, pp.339-343, (1987).

7) R. G. Melhem, "Determination of Stripe Structures for Finite Element Matrices", SIAM Journal on Numerical Analysis, vol 24, no 6, pp.1419-1433, (1987).

8) R. G. Melhem, "Toward Efficient Implementations of PCCG Methods on Vector Supercomputers", The International Journal on Supercomputer Applications, vol 1, no 1, pp.71-98, (1987).

9) D. Chiarulli, R. Melhem and S. Levitan, "Using Coincident Optical Pulses for Parallel Memory Addressing", IEEE Computer, vol 20, no 12, pp.48-58, (1987).

10) R. G. Melhem, "Verification of a Class of Self-timed Computational Networks", BIT, Vol 27, no 4 (1987), pp.480-500.

11) R. G. Melhem, "Parallel Solution of Linear Systems with Striped, Sparse Matrices", Parallel Computing, vol 6, no 2, pp. 165-184, (1988).

12) R. G. Melhem, "A Modified Frontal Technique Suitable for Parallel Systems", SIAM J. on Scientific and Statistical Computing, vol 9, no 2 (1988), pp. 289-303.

13) K. Ramarao, R. Daley and R. Melhem, "Message Complexity of the Set Intersection Problem", Information Processing Letters, vol 27, no 4, pp.169-174 (1988).

14) R. Melhem and K. Ramarao, "Multicolor Ordering of Sparse Matrices Resulting from Irregular Grids", ACM Tran. on Mathematical Software, vol 14, no 2, pp. 117-138 (1988).

15) R. Melhem and C. Guerra, "The Application of a Sequence Notation to the Design of Systolic Computations", BIT, vol 29, no 3, pp. 409-427 (1989).

16) R. Melhem, "A Systolic Accelerator for the Iterative Solution of Sparse linear systems", IEEE Trans. on Computers, vol 38, no 11, pp.1591-1595 (1989).

17) R. Melhem, D. Chiarulli and S. Levitan, "Space Multiplexing of Waveguides in Optically Interconnected Multiprocessor Systems", The Computer Journal, vol 32, no 4, pp. 362-369 (1989).

18) C. Guerra and R. Melhem, "Synthesis of Systolic Algorithm Designs", Parallel Computing, vol 12, no. 2, pp. 195-207 (1989).

19) S.P. Levitan, D.M. Chiarulli and R.G. Melhem, "Coincident Pulse Techniques for Multiprocessor Interconnection Structures", Applied Optics, vol 29, no. 14, pp. 2024-2033, (1990)

20) Y. Pan and R. Melhem, "Short Circuits in Buffered Multi-stage Interconnection Networks". The Computer Journal, vol 33, no. 4, pp. 323-329 (1990).

21) R. Melhem and G. Hwang, "Embedding Rectangular Grids into Square Grids with Dilation Two". IEEE Transactions on Computers, vol. 39, no. 12, pp. 1446-1455, (1990).

22) D. Chiarulli, S. Levitan and R. Melhem, "Optical Bus Control for Distributed Multiprocessors". The Journal of Parallel and Distributed Computing, vol. 10, no. 1, pp. 45-54 (1990).

23) M. Alam and R. Melhem, "An Efficient Spare Allocation Scheme and its Application to Fault Tolerant Binary Hypercubes". IEEE Trans. on Parallel and Distributed Systems, vol 2, no 1, pp.

117-126 (1991)

24) Z. Guo, R. Melhem, R. Hall, S. Levitan and D. Chiarulli, "Pipelined Communication in Optically Interconnected Arrays", Journal of Parallel and Distributed Computing, vol 12, no 3, pp. 269-282, (1991)

25) D. Chiarulli, R. Ditmore, S. Levitan and R. Melhem, "An All Optical Addressing Circuit: Experimental Results and Scalability Analysis", IEEE J. of Lightwave Technology, vol. 9, no. 12, pp. 1717-1725, (1991).

26) F. Provost and R. Melhem, "A Distributed Algorithm for Embedding Trees in Hypercubes with Modification for Run-time Fault Tolerance", Accepted for publication in the Journal of Parallel and Distributed Computing.

27) C. Qiao, R. Melhem, S. Levitan and D. Chiarulli, "Optical Multicasting in Linear Arrays", Accepted for Publication in the International Journal of Optical Computing.

28) R. Melhem, "Bilevel Reconfigurations of Fault Tolerant Arrays", Accepted for publication in IEEE Trans. on Computers.

PAPERS IN REFERRED CONFERENCE PROCEEDINGS: (an asterisk indicates that the paper is a preliminary version of one of the above journal papers)

1) R. G. Melhem, "A Language for the Simulation of Systolic Architectures", Proc. of The 12th. International Symposium on Computer Architecture, Boston, Mass. (1985).

2)* R. G. Melhem, "An Event Algebra for the study of Deadlock in Self-timed Computational Networks", Proc. of the 23rd Allerton Conf. on Computer, Control and Communication (1985).

3)* C. Guerra and R. Melhem, "Synthesizing Non-uniform Systolic Designs", Proc. of the International Conf. on Parallel Processing (1986).

4)* R. G. Melhem, "Application of Data-driven Networks to Sparse Matrix Multiplication", Proc. of the International Conf. on Parallel Processing (1986).

5) R. G. Melhem, "Irregular Wavefronts in Data-driven, Data-dependent Computations", Proc. of the Second Workshop on Systolic Arrays, Oxford, U.K. (1986). Also appeared in "Systolic Arrays", W. Moore, A. McCabe and R. Urquhart editors, Adam-Hilger, 1987.

6)* R. G. Melhem, "An Efficient Implementation of the SSOR/PCCG Method on Vector Computers", Proc. of the Second Int. Conf. on Supercomputers, (1987).

7)* R. G. Melhem, "Iterative Solution of Sparse Linear Systems on Systolic Arrays", Proc. of the International Conf. on Parallel Processing (1987).

8) R. G. Melhem, "Mapping Algorithms into Architectures", Proc. of the Twenty-First Annual Hawaii *International Conference on System Sciences, Vol 1 (1988).*

9) F. Provost and R. Melhem, "Fault Tolerant Embedding of Binary Trees and Rings into Hypercubes", Proc. of the International Workshop on Defect and Fault Tolerance in VLSI Systems (1988).

10)* R. Melhem and G. Hwang, "Embedding Rectangular Grids into Square Grids with Dilation Tw" Proc. of the 26rd Allerton Conf. on Computer, Control and Communication (1988).

11) M. Alam and R. Melhem, "Fault Tolerance and Reliable Routing in Augmented Hypercube Architectures", Proc. of the 8th. IEEE Phoenix Conference on Computers and Communications (1989).

12) S. Gupta and R. Melhem, "A Software Tool for the Automatic Generation of Memory Traces for Shared Memory Multiprocessor Systems", Proc. of the 22nd Annual Simulation Symposium, (1989).

13) M. Alam and R. Melhem, "How to use an Incomplete Hypercube for Fault Tolerance", Proc. of the first European Workshop on Hypercube and Distributed Computers, (1989).

14) N. Srivastava and R. Melhem, "Comparisons of Different Multistage Interconnection networks under Hot Spot Traffic Conditions", Proc. of the Twentieth Annual Pittsburgh Conference on Modeling and Simulation, (1989).

15)* R. Melhem, "Bi-Level Reconfigurations of Fault Tolerant Arrays in Bi-modal Computational Environments", Proc. of the 19th. International IEEE Symposium on Fault Tolerant Computing (1989).

16) Z. Guo and R. Melhem, "Embedding Pyramids in Array Processors With Pipelined Busses", Proc. of the International Conf. on Application Specific Array Processors, September (1990).

17)* Z. Guo, R. Melhem, R. Hall, S. Levitan and D. Chiarulli, "Array Processors with Pipelined Optical Busses", Proc. of the Frontiers 90 Conference on Massively Parallel Computation, (1990).

18) T. Znati and R. Melhem, "Personalized Distributed Systems", Proc. of the ISMM International Conference on Parallel and Distributed Computing and Systems, (1990).

19)* Z. Guo, R. Melhem, R. Hall, S. Levitan and D. Chiarulli, "Pipelined Communications on Optical Busses", Proc. of the SPIE International Symposium on Advances in Interconnections and Packaging", (1990).

20)* D. Chiarulli, S. Levitan and R. Melhem, "Self Routing Interconnection Structures Using Coincident Pulse Techniques", Proc. of the SPIE International Symposium on Advances in Interconnections and Packaging", (1990).

21)* D. Chiarulli, S. Levitan and R. Melhem, "Demonstration of an All Optical Addressing Circuit", Technical Digest of the Optical Computing Topical Meeting, Salt-Lake city, (1991).

22) R. Melhem and John Ramirez, "Reconfiguration of Computational Arrays with Multiple Redundancy", Proc. of the International Conference on Parallel Processing, (1991).

23) M. Alam and R. Melhem, "Channel Multiplexing in Modular Fault Tolerant' Multiprocessors", Proc. of the International Conference on Parallel Processing, (1991).

24)* C. Qiao, R. Melhem, S. Levitan and D. Chiarulli, "Multicasting in Optical Bus Connected Processors *Using Coincident Pulse Techniques*", Proc. of the International Conference on Parallel Processing, (1991).

25) T. Znati, K. Pruhs and R. Melhem, "Dilation Based Bidding Schemes for Dynamic Load Balancing on Distributed Processing Systems", Proc. of the sixth Distributed Memory Computing Conference, (1991).

26) R. Melhem, K. Pruhs and T. Znati, "Using Spanning Trees for Balancing Dynamic Load on Multiprocessors", Proc. of the sixth Distributed Memory Computing Conference, (1991).

27) R. Melhem and John Ramirez, "Meshes with Flexible Redundancy", Proc. of the second Workshop on Algorithms and Parallel VLSI Architectures, Bonas, France, (1991).

28) F. Provost and R. Melhem, "Embedding Rings in Hypercubes for Run-time Fault Tolerance", Proc. of the fourth ISMM Conference on Parallel and Distributed Computing and Systems, (1991).

29) A. Varvitsiotis, S. Theodoridis and R. Melhem, "Mapping FIR Filtering on Systolic Rings", Proc. of the International Conf. on Application Specific Array Processors, September (1991).

30) N. Shrivastava and R. Melhem, "Efficient and Optimal Fault-to-Spare Assignment in Doubly Fault Tolerant Arrays", Proc. of the IEEE Int. Workshop on Defect and Faults Tolerance in VLSI Systems, (1991).

31) C. Qiao, R. Melhem, "Time-division Optical Communications in Multiprocessor Arrays", Proc. of the Supercomputing 91 conference, IEEE Press (1991).

6.2 Students Funded During Current Period

- Zicheng Guo (Ph. D.) Thesis Title: Array Processors with Pipelined Busses and Their Implication in Optically and Electronically Interconnected Multiprocessor Architectures. Expected completion date: April, 91. Supported: Summer of 1989, and all of 1990.
- Chunming Qiao (Ph.D.) Expected completion date: April 1993. Supported: September 1990 - present.
- David George (M.S.) April 1991. Topic: Synthesis of Asynchronous Finite State Machines. Supported: Summer of 1990.
- Tom George (M.S.) April 1991. Topic: Extended Simulation Models for VHDL. Supported: Summer of 1990.
- James Tezza (B.S.) (M.S) Expected completion date: August 1992. Topic: Power Distribution in Lossless Tapped Erbium Doped Fiber Busses for Multiprocessors. Supported: Spring 1991 - present.
- Manoj Bidnurkar (M.S.) Expected completion date: August 1992 Topic: Computer Model of Erbium Doped Fiber. Supported January 1992 - present.

7 Project Interactions

7.1 Conferences and Workshops

- Chiarulli, Guo, and Levitan attended and presented at SPIE Boston/OE'90, November 1991, Boston, MA.
- Guo and Melhem attended and presented at Frontiers'90 Conference, October 1990, College Park, MD.
- Guo and Melhem attended and presented at Intl. Conf. on Application Specific Array Processors, September 1990, Princeton, N.J.
- Levitan and Chiarulli organized and chaired a panel discussion on "The Future of Optics in Computing" at the November 1991 Supercomputing Conference. Qiao also presented a paper at that conference.
- Melhem and Qiao attended and presented at the International Conference on Parallel Processing, August, 1991.

7.2 Invited Presentations

- Chiarulli and Levitan gave invited talks at University of Colorado at Boulder, Optoelectronic Computing Systems Center, Boulder, CO, February 6, 1990.
- Members of the group have been invited to participate in an AFOSR sponsored workshop on Reconfigurable Optical Interconnects, Boulder, CO, March, 1992.

7.3 Other Interactions

- Chiarulli and Melhem are Guest editors for a planned special issue of the Journal of Parallel and Distributed Computing on Optical Computing.
- Levitan is on the Ph.D. committee's of Brian Telfer, Sanjay Natarajan and John-Scott Smokelin, students of Prof. David Casasent from Carnegie Mellon University.
- Melhem is the program chair, and Levitan is on the program committee of the Fifth ISSM International Conference on Parallel and Distributed Computing and Systems, to be held in Pittsburgh, October, 1992.
- Levitan gave talks about the group's work at the IBM T.J. Watson Research Center (October 1991), and to the Department of Electrical Engineering at the University of Pittsburgh (January 1991).
- Chiarulli ran an Optical Computing graduate seminar in the Department of Computer Science.
- Richard Thompson has agreed to be on the Ph.D. committee of Chunming Qiao. Professor Thompson also gave a talk at the Department of Electrical Engineering.
- We have been interacting with Dr. William Miniscalco from GTE Labs on our work with Erbium doped glass fiber. GTE has given us samples of doped fiber for our experiments.

8 Project New Discoveries

- We have quantified the advantages and general applicability of signal pipelining for interconnection networks.
- We have resolved (from both a theoretical and a practical point of view) the issue of shadows in multi-dimensional structures.
- We have realized the generalization of signal pipelines to TDM structures and further to SDM and WDM based networks as well.

9 Project Evaluation

We are pleased with the accomplishments of the group to date. We have made significant progress with regards to our fabrication of prototype structures to verify the applicability of our ideas to multiprocessor interconnection networks. Our latest work in the laboratory on lossless tapped structures is promising.

We believe that our generalization of coincident structures to pipelined structures, and pipelined structures to more general reconfigurable structures will be a significant contribution to the theory and practice of high speed multiprocessor interconnection networks.

A Copies of Recent Papers from the Research Group

1. An all Optical Addressing Circuit: Experimental Results and Scalability Analysis *IEEE Journal of Lightwave Technology 9:12, 1991*

2. Optical Multicasting in Linear Arrays *International Journal on Optical Computing, in press*

3. Pipelined Communications in Optically Interconnected Arrays *Journal of Parallel and Distributed Computing, 12:3, 1991*

4. Time-Division Optical Communications in Multiprocessor Arrays *Supercomputing'91, Proceedings*

An All Optical Addressing Circuit: Experimental Results and Scalability Analysis

Donald M. Chiarulli, *Member, IEEE*, Robert M. Ditmore, Steven P. Levitan, *Member, IEEE*, and Rami G. Melhem, *Member, IEEE*

***Abstract*—In this paper, we present results from a demonstration of both single and parallel selection in a one of four optical addressing circuit operating at 250 MHz using coincident pulse addressing. We then present an analysis of power distribution in two different tapped fiber structures. Based on our results, we discuss issues of scalability with respect to synchronization and power distribution in larger systems.**

I. Introduction

TWO properties of optical signals, unidirectional propagation and predictable path delay, make it possible to devise logic systems in which information is encoded as the relative timing of two optical signals. Coincident pulse addressing is an example of such a system. In this technique, the address of a detector site is encoded as the delay between two optical pulses which traverse independent optical paths to the detector. The delay is encoded to correspond exactly to the difference between the two optical path lengths. Thus, pulse coincidence, a single pulse with power equal to the sum of the two addressing pulses, is seen at the selected detector site. Other detectors along the two optical paths for which the delay did not equal the difference in path length, detect both pulses independently, separated in time.

Stated more formally, consider an optical fiber of length L with two optical pulse sources, P_1 and P_2 coupled to each end. Each source generates pulses of width τ and height h. Define $l = \tau c_g$ where c_g is the speed of light in the fiber. In other words l is the length of fiber corresponding to the pulse width. Using 2×2 passive couplers, n detectors, labeled D_1 through D_n, are placed in the fiber with the two tap fibers from each coupler cut to equal lengths and joined at the detector site. The location of each coupler/detector is carefully measured so that the ith detector is located at $(L - (n + 1)l)/2 + il$, from the left end of the bus. The optical bus in the center of Fig. 1 shows such an arrangement for $n = 3$. To uniquely address any detector, a specific delay between the pulses generated by P_1 and P_2 is chosen. If this delay is $(n - 2i + 1)\tau$, the two pulses will be coincident at detector D_i.

The same technique can be generalized to support parallel selections. If the P_1 source generates a single pulse at time t_r and the source P_2 generates a series of pulses at times t_i, $i \in \{1, \cdots, n\}$ with each t_i timed relative to t_r, then, according to the addressing equation above, to select a specific detector i each t_i will be in the range $-(n - 1)\tau \leq t_r - t_i \leq (n - 1)\tau$. Therefore, any or all of the i detectors can be uniquely addressed by a positionally distinguishable pulse from source P_2. For convenience, this pulse train is referred to as the select pulse train and the single pulse emanating from P_1 is called the reference pulse. Since the length of the select pulse train is n, and each pulse in the return to zero encoding is separated by 2τ, it follows that the system latency, $\sigma = 2n\tau$. Further, up to n locations may be selected in parallel within a single latency period. Therefore, the system throughput is $\nu = 1/2\tau$.

In previous papers, we have discussed the general application of coincident pulse techniques to both memory addressing and multiprocessor network applications [2], [3], [7], [8]. In this paper, we emphasize the practical limits on the applicability of this technique for large systems. In order to design large scale computer systems, we need to know the realistic limits on the speed, size, and cost of such systems. Our long term goal is to build high-speed multiprocessor interconnection networks using off the shelf optical components and tapped fiber busses.

Tapped fiber busses, those with one or more transmitter and multiple receivers, have been less widely adopted than simple point-to-point fibers, primarily because of scalability limits based on power distribution [9]. However, the recent development of low ratio passive couplers [5] and the prospect of fiber based optical amplifiers [4], [6] suggest a closer examination of the power distribution problem. Therefore, we have constructed a prototype system for conducting experiments from which we can extrapolate reasonable limits on the speed and size of practical multicomputer systems.

In this paper, we first present results from two laboratory experiments on a prototype coincident pulse addressing system. The two questions to be answered by the experiments are: how do synchronization error and power loss effect the scalability of such systems. Therefore, the first experimental is an examination of the coincident pulse

Manuscript received March 25, 1991, revised July 23, 1991. This work was supported, in part, by the Air Force Office of Scientific Research under Grant number: AFOSR-89-0469.

D. M. Chiarulli and R. G. Melhem are with the Department of Computer Science, University of Pittsburgh, Pittsburgh, PA 15260.

R. M. Ditmore is with Intergraph Corporation, Huntsville, AL.

S. P. Levitan is with the Department of Electrical Engineering, University of Pittsburgh, Pittsburgh, PA 15261.

IEEE Log Number 9103633

0733-8724/91$01.00 © 1991 IEEE

power, as a function of the synchronization of the arriving pulses. The second experiment demonstrates our ability to select arbitrary subsets of the detectors with a select pulse train operating at 250 MHz. The ability to perform selections of multiple detectors is key to various computer system applications we have investigated. However, the second experiment highlights a more fundamental problem: the power loss due to the tapping couplers on the bus diminishes the ratio of coincident to noncoincident pulse heights for long bus structures. The two experiments contrast the temporal and physical scalability of coincident pulse systems and show that the dominant effects are, in fact, power distribution limits on the physical scale of these systems.

Section III expands on the power distribution issue with an analysis of power distribution in two tapped fiber network structures. The first is the same linear structure that we use in our experiments. The second is a dual-level structure that consists of a main fiber and a series of secondary distribution fibers from which power is tapped. We conclude with a discussion of the implications of these findings to the construction of large systems.

II. Experimental Results

Fig. 1 is a diagram of the prototype structure. The fiber bus consists of a length of multimode fiber tapped three times using Gould 10-dB fiber couplers. Select and reference bit patterns are generated by modulating the 4-ns pulse output of a Tektronix PG502 pulse generator, shown in the diagram as clock, with the output of two ECL shift registers, one for select, one for reference, at gates G2 and G3. Gates G1 and G4 simultaneously hold the diode current for laser diodes P1 and P2 respectively at threshold, while the outputs of G2 and G4 generate modulation current. The result is two, 4-bit, return to zero bit streams, which encode the information in each of the shift registers. As explained above, this allows us to select any subset of the three (and in later experiments four) detectors. The use of two shift registers allows us flexibility in the positioning of the reference pulse relative to the select pulse train.

A. Pulse Synchronization

In our first experiment, measurements were made to characterize the effect of synchronization error between the reference and select pulses on the power of the coincident pulse. Since this error can be characterized as a percentage of the pulse width, synchronization precision has a direct bearing on the absolute width and height of an addressing pulse that can be effectively detected.

A coincident pulse structure with three detectors was used, as shown in Fig. 1. This allowed detector D_2 to be located in the center of the bus resulting in exactly equal noncoincident pulse heights, as shown in the oscilloscope trace of Fig. 2.

The reference and select pulse trains were configured to select D_2. In each step of the experiment, synchronization error was introduced by adding successively longer lengths of fiber to the ends of the bus. Length was added first on the reference pulse end of the bus, and then on the select pulse end of the bus.

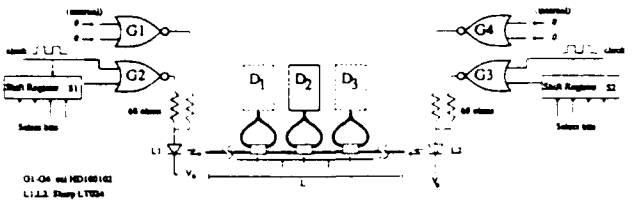


Fig. 1. Synchronization experiment.

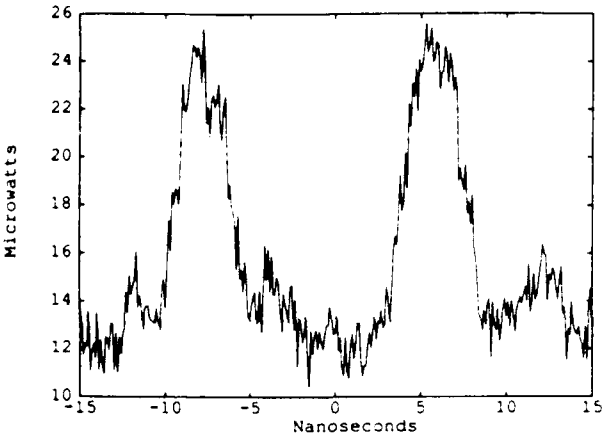


Fig. 2. Synchronization experiment waveform.

Fig. 3 shows the reduction factor, f, of the coincident pulse power as a function of percent synchronization error. Percent synchronization error is the error, in time units, introduced by each length of fiber divided by the pulse width. In other words, pulses at perfect coincidence (synchronization error = 0) yield a reduction factor of $f = 1.0$, which implies a coincident power equal to twice the single pulse power.

Synchronization error in either the select pulse, shown as positive error, or the reference pulse, shown as negative error, reduces this power by the factors shown. The solid line in Fig. 3 is the experimental result. The dotted line is an analytical result generated from the coincidence of two sinusoidal pulse waveforms. In both cases, the power falls off in roughly the shape of the coincident waveforms themselves.

In order to analyze this result, we must consider the sources of synchronization error. Assuming that manufacturing tolerances for electronic components and errors in fiber length measurements can be compensated for by tuning the system, the primary sources of synchronization error will be thermal variations in both the optical characteristics of the fiber and in the performance of electronic components as well as any jitter introduced by the electrical clock generators. For the former, recent results [10] have shown that the variability of the index of refraction of the fiber versus temperature is on the order of 40 ps/km-degree C, and that this is the dominant tempera-

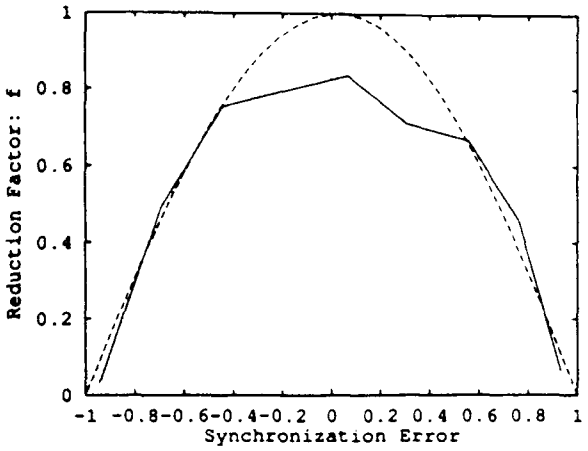


Fig. 3. Synchronization error reduction factor.

ture effect. This represents a very minor variation in effective optical path length. Obviously, jitter and thermal effects in the electronics will be the predominant sources of synchronization error.

However, from Fig. 3 we can see that a timing synchronization error of up to 50% only decreases the coincident pulse power to about 70% of its ideal value. Therefore, large variations (on the order of one half of a pulse width) in electronic pulse generation can be tolerated without significant degradation of the coincident signal. This result characterizes a temporal limit on scalability, based on a limit of achievable pulse widths. Timing errors of several hundred picoseconds are tolerable in gigahertz systems. Therefore, using off the shelf components operating in the one gigahertz range, $\tau = 1$ ns and a system throughput of $\nu = 1/2\tau = 500 \times 10^6$ addressing operations per second is feasible.

The other primary limit, which we need to address, is optical power distribution. Since we are using a passive bus structure, the optical signals are not amplified at any point on the bus. Therefore, sufficient optical power must be available at each detector to individually discriminate coincidence from noncoincidence in the presence of selection pulses for other detectors and noise. This is the subject of the second experiment.

B. Coincident Pulse Power

Our second set of experiments were used to characterize the effect of detector position on the available coincident pulse power. A similar experimental setup was used, this time with four detectors, as shown in Fig. 4.

Figs. 5–8 show the output waveforms for detectors D1 and D3 for various selection patterns. Note that for each selection pattern (pair of waveforms) the experimental equipment was adjusted so that the absolute values of pulse heights for different selection patterns varied. Figs. 5 and 6 show coincident and noncoincident waveforms at detectors D1 and D3, respectively. Note that in both cases, the noncoincident waveforms (shown in (b)) are of unequal power. This is due to the fact that each pulse has passed through a different number of couplers and, hence,

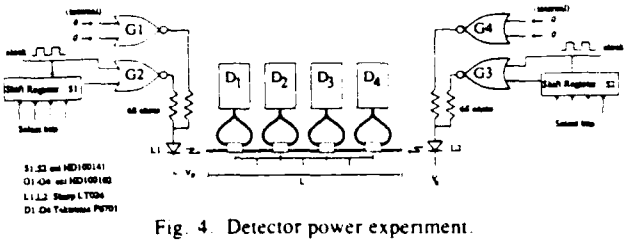


Fig. 4. Detector power experiment.

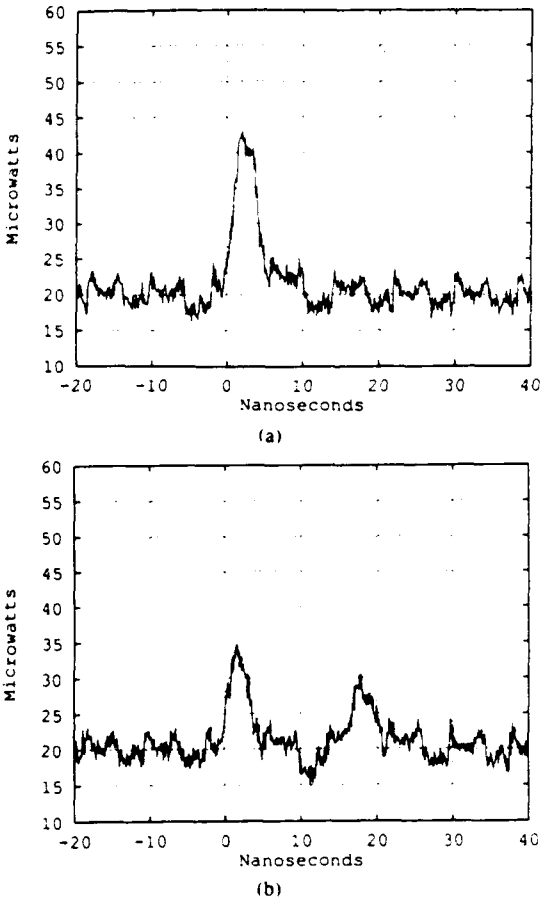


Fig. 5. (a) Selection of D1 measured at D1. (b) selection of D3 measured at D1.

has become attenuated to different levels. This clearly shows that the relative power between coincident and noncoincident pulses is a function of the detector location.

Figs. 7 and 8 are examples of parallel selections. The waveform in Fig. 7(a) shows a parallel selection waveform at detector site D_3 for the selection of three detectors, including D_3. This incident waveform peak is comparable to the noncoincident waveform, in Fig. 7(b), in which D_3 has been removed from the set of selected locations. Similarly Fig. 8 shows parallel selection of all four detectors at sites D_1 and D_3.

To quantify the power degradation that we observed in these experiments, we define the amount of additional power in a coincident pulse relative to the largest noncoincident pulse seen by a detector as the power margin, *Pm*. This is given as a fraction of the maximum noncoincident pulse power:

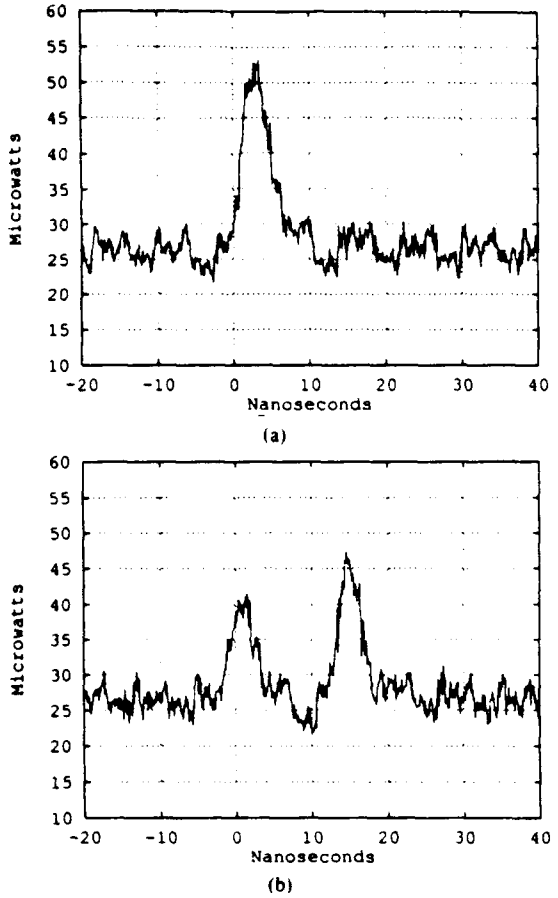


Fig. 6. (a) Selection of D3 measured at D3, (b) selection of D1 measured at D3

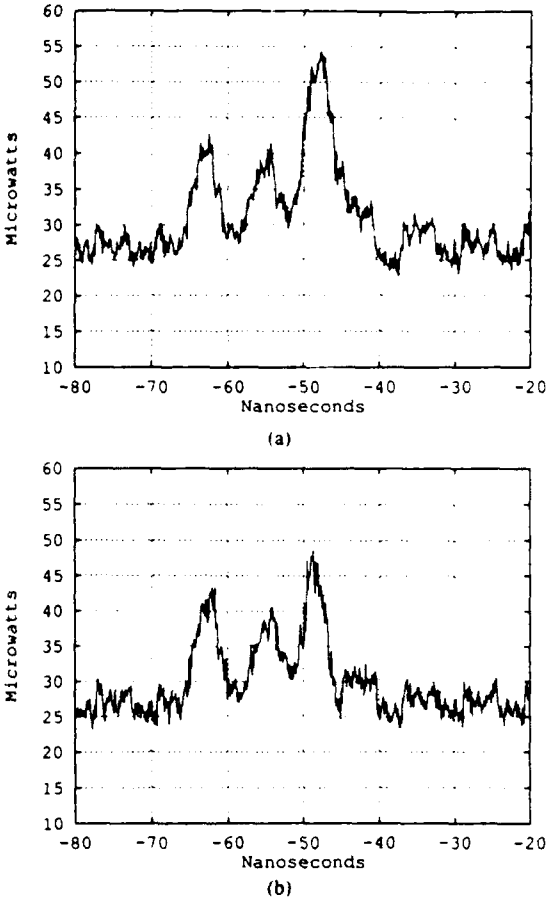


Fig. 7. (a) Selection of D1, D2, D3 measured at D3, (b) selection of D1, D2 measured at D3.

$$Pm = (p_1 + p_2 - \max(p_1, p_2))/\max(p_1, p_2)$$
$$= \min(p_1, p_2)/\max(p_1, p_2). \quad (1)$$

Pm indicates the threshold level needed for a detector to discriminate between coincident and noncoincident pulses. That is, for each detector on the bus the threshold should be set to be at:

$$((Pm + 1) \times \max(p1, p2))/2.$$

Pm has its maximum value, $Pm = 1$, at the center of the bus, where each pulse is at equal power, and coincidence is reflected as a doubling of power seen by the detector. It is at its minimum value at the ends of the bus. For all the selection experiments shown in Figs. 5 through 8, the power margin is in excess of 30%. That is, coincident power is greater than 130% of peak single pulse power. This is measured at D_1, which is the leftmost detector on the bus.

In the next section we discuss the implications of power margin on scalability issues.

III. Analytical Study of Power Distribution

In this section, we present an analysis of power distribution in each of two tapped fiber network structures. The first is a simple linear structure with a single backbone and a series of passive coupler taps, as used in the experiment above. The second is a dual level structure, which consists of a backbone fiber and a series of secondary distribution fibers from which power is tapped.

In this analysis, we use passive, bidirectional, 2 × 2, symmetric fiber couplers as shown in Fig. 9 [1], [5]. These are identical to the couplers we used in our previously discussed experiments, except that in our analysis we assume no excess loss in the couplers. Since the couplers are bidirectional, we arbitrarily let A, B be the input ports and A', B' be the output ports. Equation (2) shows power distribution from the input to the output:

$$\begin{pmatrix} A' \\ B' \end{pmatrix} = \begin{pmatrix} r & (1-r) \\ (1-r) & r \end{pmatrix} \begin{pmatrix} A \\ B \end{pmatrix} \quad (2)$$

where r is the coupling ratio. Using these couplers, we now discuss the linear and dual level structures.

A. The Linear Structure

As is shown in Fig. 10, a linear bus consists of n detectors (and n couplers). Assuming two, unit height, pulses starting at opposite ends of the bus, and one type of coupler with a ratio of r, the optical power from each pulse p_i^1 and p_i^2 at detector D_i is given by the equations:

$$p_i^1 = r^{(i-1)}(1-r), \qquad p_i^2 = r^{(n-i)}(1-r). \quad (3)$$

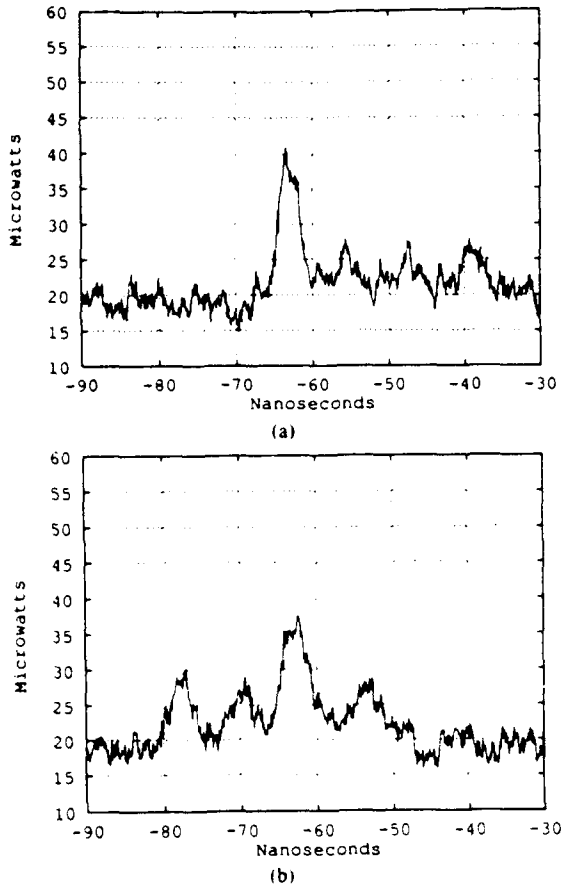


Fig. 8 (a) Selection of D1, D2, D3, D4 measured at detector D1, (b) selection of D1, D2, D3, D4 measured at detector D3

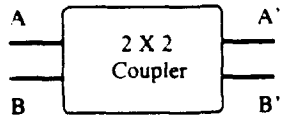


Fig. 9. Symmetric fiber coupler.

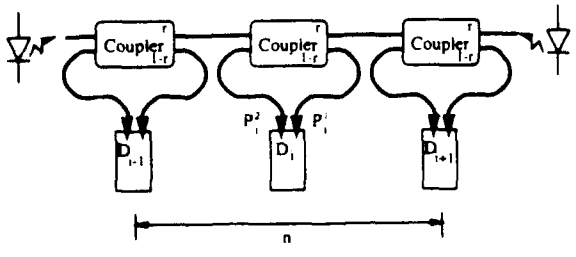


Fig. 10. Linear optical bus.

Since the bus is symmetrical, we can analyze one signal that originates on the left from a single transmitter and propagates to the right as shown in Fig. 10.

Fig. 11 is a plot of $p_i^!$ versus i for various values of r. Note that the values of i are plotted on a logarithmic scale. The topmost curve is for a bus with $r = 90\%$ where the power at the first detector is 10% of the initial power. The lowest curve is for a bus with $r = 99\%$ where power at the first detector is 1% of the initial pulse power. For all

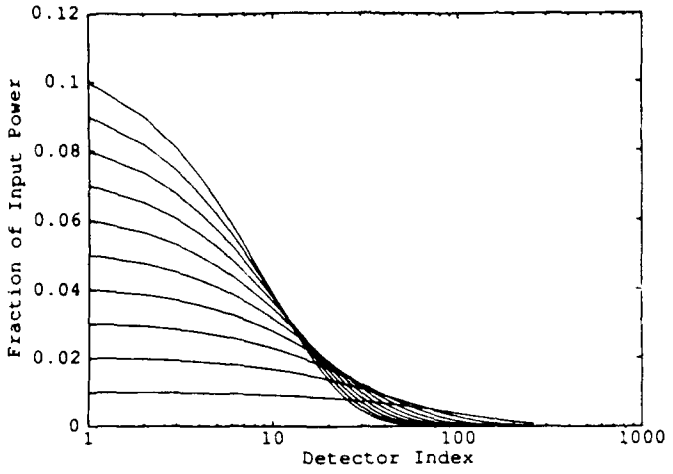


Fig. 11. Power $p_i^!$ at detector D_i for $90\% \le r \le 99\%$.

the curves, the absolute power falls off geometrically with increasing i, $1 \le i \le n$.

A bound on the number of detectors, n is determined by the sensitivity of the last detector on the bus. In other words, it is the bound for a detector to discriminate between "no pulse" and "pulse." If the last detector has a sensitivity $P\text{min}$, then the maximum number of detectors supportable is

$$n = \frac{\log\left(\frac{P\text{min}}{1-r}\right)}{\log(r)} + 1. \tag{4}$$

Equation (4) is shown graphically in Fig. 12 for a set of coupling ratios $r = 90\%, 95\%, 97\%, 98\%, 99\%$, and $0.01\% \le P\text{min} \le 1\%$ of the input power on a logarithmic scale. This graph confirms the intuition that by improving either the coupling ratio r, or the sensitivity of the detectors $P\text{min}$, we will be able to support more detectors on the bus. We also note the sharp drop in n for high values of $P\text{min}$ and r, which reflects the situation where much of the available power flows off the end of the bus and is wasted.

However, for our experimental setup, it is clear that it is not the absolute power but rather the power margin that imposes a bound on the size of the system. In addition, since the bus configuration chosen for this structure requires bidirectional propagation, we are constrained to use a single tapping ratio, r, for all couplers. Based on these two constants, the graph shown in Fig. 13, which is a plot of worst-case power margin Pm versus $1 - r$ for various bus lengths, confirms that the power margin for the coincident structure bounds scalability more strongly than absolute power. We can see from Fig. 12 that using commercially available 95% couplers, and assuming we can tolerate a $P\text{min}$ of 0.0001 of input power we could achieve bus lengths of abut 120 detectors. This would be the case for an input of 100 mW of power injected into the bus, and a detector sensitivity of 10 μW, operating at 250 MHz. However, Fig. 13 shows that for a power margin of $Pm = 20\%$ we could only reach lengths of 32 detec-

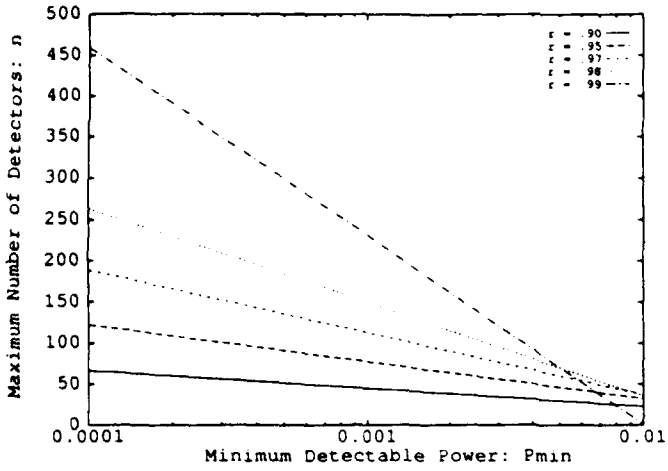


Fig. 12. Number of detectors versus *Pmin* for various values of *r*.

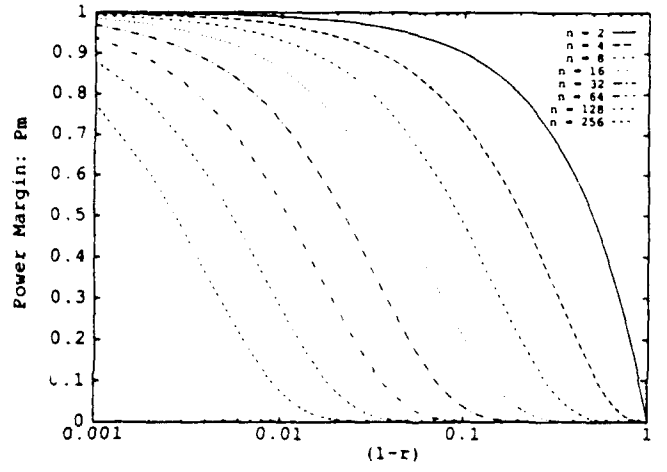


Fig. 13. Power margin *Pm* versus $0.001 \geq (1 - r) \geq 1$ for various bus sizes.

tors. Therefore, due to both minimum power constraints and power margin issues the system scale is highly sensitive to the fixed value of *r*. Further, we note that power margin imposes a tighter constraint than absolute power.

To help alleviate this problem, we propose a two level bus structure. By using two levels, we can essentially increase the tapping ratios on our buses and more effectively control the amount of power at each detector.

B. The Dual-Level Structure

The basis for the power distribution problem in the linear system is the fact that detectors at the start of the bus use more power than needed and, therefore, detectors at the end of the bus are starved. If we were to relax the requirement of fixed ratio taps in favor of varying the coupling ratios, we would need a number of distinct, precisely tuned couplers approaching the number of detector sites [9]. Yet, no couplers exist that would allow tuning to a precision of more than one or two percent. Of course, the use of tuned couplers forces the network to be unidirectional since coupling ratios must decrease in the direction of propagation.

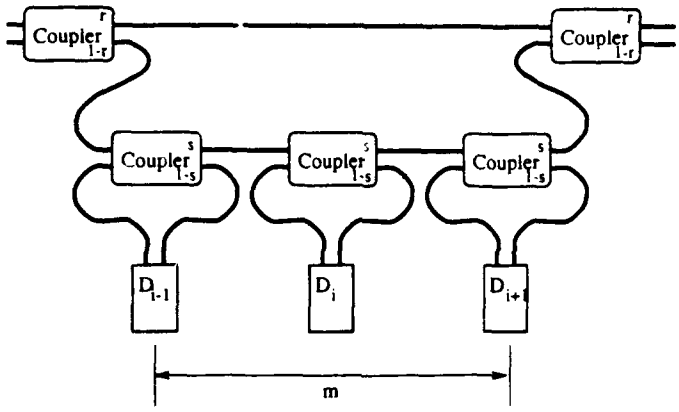


Fig. 14. Dual-level optical bus.

An alternative method that does not require multiple coupling ratios is to adopt a dual-level bus structure. As shown in Fig. 14, we split the bus into a main fiber and a sublevel to create a section of the bus, labeled *m*. The sublevel contains *m* detectors in a linear arrangement except for the last detector, which feeds back the remaining power into the main fiber and the next section. In the main fiber, care must be taken to ensure that the optical path length is the same as the subsection so that the two parts of the signal arrive synchronized at the next section. The dual-level bus consists of a series of these sections.

Once again, we start with an analysis of absolute power for this structure, and then proceed to power margin issues. Thus, we assume the input is from the left (into the upper leg to the first coupler) and propagates to the right. The detectors are numbered linearly in the direction of propagation.

We further assume two types of couplers with splitting ratios of *r* and *s* for the main level and sublevel, respectively. The power at any given detector site in Fig. 14 is given by

$$p_i = (1 - r \quad r)A^k \begin{pmatrix} 1 \\ 0 \end{pmatrix} s^l(1 - s) \tag{5}$$

where p_i is the power at site i, r, and s are coupling ratios, k is i div m, l is i mod m, m is the number of detectors in a sublevel, and

$$A = \begin{pmatrix} r & 1 - r \\ (1 - r)s^m & rs^m \end{pmatrix}. \tag{6}$$

From linear algebra [11], we know that a vector of the form $u_k = A^k u_o$ can be rewritten as $u_k = \sum_{i=1}^{n} c_i \lambda_i^k x_i$, where λ_i are the eigenvalues of matrix A, the x_i's are the associated eigenvectors and the coefficients c_i are determined from the initial condition u_o.

For our analysis, we rewrite the matrix of (5) in the form

$$A^k \begin{pmatrix} 1 \\ 0 \end{pmatrix} = c_1 \lambda_1^k x_1 + c_2 \lambda_2^k x_2 \tag{7}$$

and the coefficients are determined by $c_1 x_1 + c_2 x_2 = u_o$.

The x_i are vectors and they are given by $x_i = \binom{\xi_i}{1}$. Rewriting the coefficient equation gives

$$\begin{pmatrix} c_1\xi_1 \\ c1 \end{pmatrix} + \begin{pmatrix} c_2\xi_2 \\ c2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

which has the solution

$$c_1 = \frac{1}{\xi_1 - \xi_2}.$$

Assuming, without loss of generality, that $\lambda_1 > \lambda_2$ as k increases, then the $c_1\lambda_1^k x_1$ term in (7) quickly dominates. Therefore, a good approximation is given by

$$p_i = [(1 - r) \quad r]c_1\lambda_1^k x_1 s^i(1 - s). \tag{8}$$

Fig. 15 shows a comparison of a linear bus and a dual-level structure for the particular case of $r = s = 90\%$, $n = 256$, and $m = sqrt(n)$. Clearly, the power at the detectors for the linear bus falls off much more rapidly than for the dual-level bus. The dual-level bus shows a characteristic "saw-tooth" pattern of power distribution. At the beginning of each section, power is restored by injection of power from the main backbone. This more evenly distributes all of the available power down the length of the bus.

In the linear structure, we examined the bounds for the minimum power needed at the last detector. For the dual-level structure, we will examine the minimum power seen at the last detector of the last section. This minimum power is given by the equation

$$Pmin = \lambda_1^k(\xi_1(1 - r) + r)c_1 s^{m-1}(1 - s). \tag{9}$$

As with the linear case, the ability to support large systems is dependent upon maximizing the values of r and s. However, in the dual-level case, we additionally may vary m, the number of detectors per section. The relationship between r, s, and m is captured in λ_1, which is a monotonically increasing function of r, and s but is not monotonic in m. Therefore, it is desirable to fix r and s to be as large as possible and adjust m to maximize the total number of detectors in the system.

This relationship is shown in Fig. 16. The two families of curves represent coupling ratios of $r = s = 90\%$ and $r = s = 95\%$. The curves are the number of detectors (length of the bus) supportable at different Pmin values. For the 90% curves, Pmin = 0.0001, 0.0002, 0.0004, 0.0008, 0.0016, 0.0032, and 0.0064. For the 95% curves, Pmin = 0.0001, 0.0002, 0.0004, 0.0008, and 0.0016. Note that the dual-level structure with 95% couplers cannot support high minimum power detectors since the power into the first detector $p_1 = 0.05 \times 0.05 = 0.0025$. The long tails on the curves reflect the condition where $m \geq n$.

Having chosen values for r, s, and m, we can rewrite equation (9) to compute the number of detectors supportable as a function of Pmin:

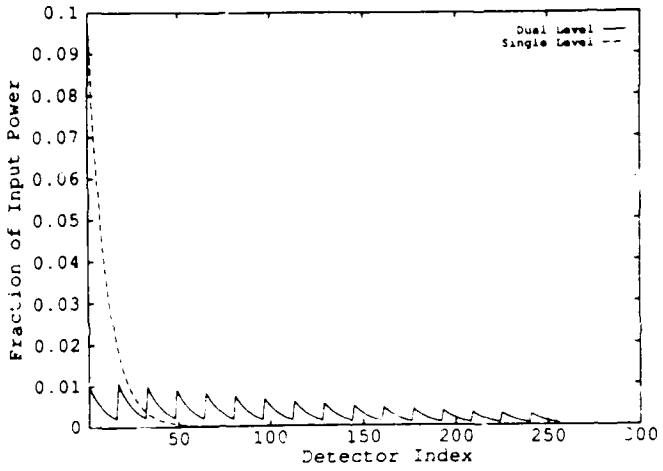


Fig. 15. Power at detector sites for single- and dual-level 256 node buses

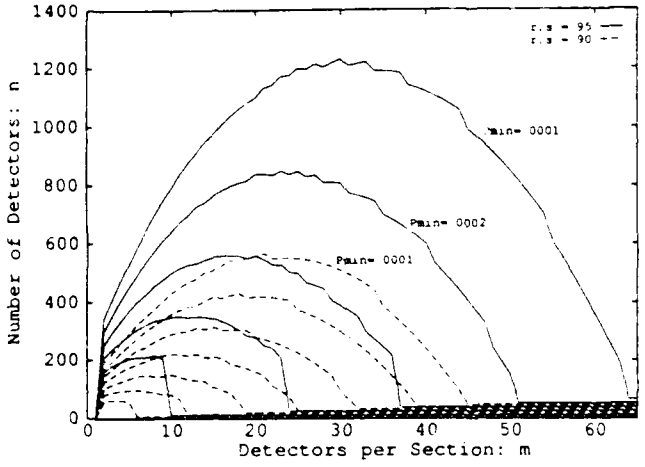


Fig. 16. n versus m for r, $s = 0.90$, 0.95, and various Pmin.

$$n = m \frac{\log\left(\dfrac{Pmin}{((1 - r)x_1 + r)c_1 s^{k-1}(1 - s)}\right)}{\log(\lambda_1)}. \tag{10}$$

A plot of numerical solutions for (10) is shown in Fig. 17.

Equation (10) and Fig. 17 allow a direct comparison of the dual-level bus performance shown in Fig. 17 with linear bus performance derived in (4) and plotted in Fig. 12. From this comparison, we can see that, in terms of Pmin, the optimized dual level bus gives approximate factors of between 4 and 10 improvement (depending on the coupler ratios) over the simple linear configuration

To perform the analysis of power margin for the two level structure, we compare the maximum power at any detector to the minimum power at any detector on the bus. This simplifies the calculation and gives a bound on the "envelope" of the saw-tooth power curve (as shown in Fig. 15). For these curves (shown in Fig. 18) we are again using $m = sqrt(n)$ and $r = s$. Unlike the curves for the linear bus, these curves have a peak and approach an asymptotic value for very large values of r and s. This is

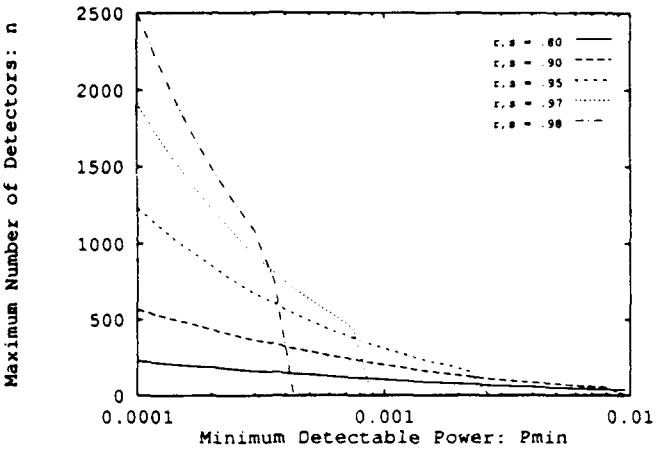


Fig. 17. *P*min versus *n* for different values of *r*, *s*.

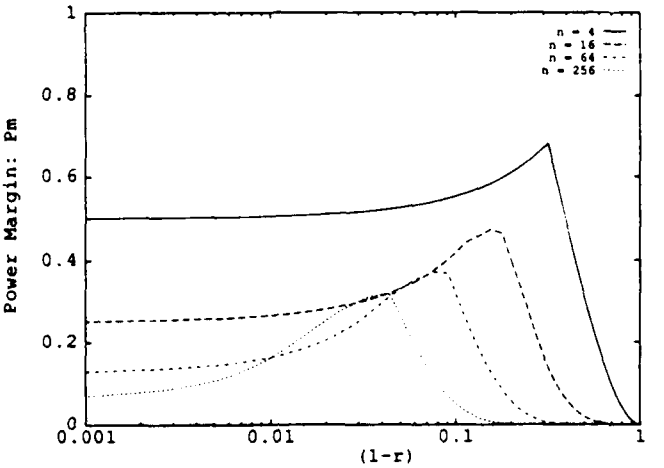


Fig. 18. *Pm* versus $0.001 \geq (1 - r) = (1 - s) \geq 0$ for dual-level buses.

because, similar to the linear case, we reach a point at which a significant percentage of the power must be thrown away at the end of the bus, in order to account for the large coupling ratio of the final tap. However, it is still the case that *Pm*, the power maximum margin, limits the scalability of the system more tightly than absolute power. As a practical example similar to the linear case, using available 95% percent couplers, the power margin limits bus size to about 300 detectors, rather than the 1250 detectors we could expect based on minimum power requirements of *P*min = 0.0001.

IV. Summary

Clearly, three factors, threshold power margin, synchronization error, and coupling ratio determine system scale. Our experiments have shown that the important system issues of latency and throughput which are related to pulse width limits are highly scalable. Based on current and near term technology, we have shown that synchronization error does not contribute significantly to the bounds calculated above.

On the other hand, physical scalability issues such as the size of the bus and the number of detectors that can be supported are more severely restricted due to power distribution in a system built from passive couplers. However, we believe near term technologies (e.g., fiber amplifiers) and alternate bus structures will alleviate this problem. The fact that the temporal scalability bounds show significantly shorter pulses can be supported, is very encouraging for the long-term application of this technique.

References

[1] F. C. Allard, *Fiber Optics Handbook For Engineers and Scientists*. New York: McGraw-Hill, 1990.

[2] D. M. Chiarulli, S. P. Levitan, and R. G. Melhem, "Self routing interconnection structures using coincident pulse techniques," in *SPIE OE/Boston '90* (Boston, MA), Nov. 4-9, 1990.

[3] D. M. Chiarulli, R. G. Melhem, and S. P. Levitan, "Parallel memory using coincident optical pulses," *IEEE Comput.*, vol. 20, no. 12, pp. 48-57, Dec. 1987.

[4] C. R. Giles, E. Desurvire, J. R. Talman, J.R. Simpson, and P. C. Becker, "2-Gb/s signal amplification at $\lambda = 1.53$ μm in a erbium-doped single-mode fiber amplifier," *J. Lightwave Technol.*, vol. 7, no. 4, pp. 651-656, Apr. 1989.

[5] Gould Electronics, Glenn Burnie, MD, Gould Fiber Optics Technical Notes.

[6] R. I. Laming *et al.*, "Efficient pump wavelengths of erbium-doped fiber optical amplifiers," *Electron. Lett.*, vol. 25, no. 1, pp. 12-14, Jan. 1989.

[7] S. P. Levitan, D. M. Chiarulli, and R. C. Melhem, "Coincident pulse techniques for multiprocessor interconnection structures," *Appl. Opt.*, vol. 29, no. 14, pp. 2024-2033, May 1990.

[8] R. G. Melhem, D. M. Chiarulli, and S. P. Levitan, "Space multiplexing of optical waveguides in a distributed multiprocessor," *Comput. J. British Comput. Society*, vol. 32, no. 4, pp. 362-369, 1989.

[9] M. Nassehi, F. Tobagi, and M. Marhic, "Fiber optic configurations for local area networks," *IEEE J. Select. Areas Commun.*, vol. SAC-3, no. 6, pp. 941-949, Nov. 1985.

[10] D. Sarrazin, H. Jordan, and V. Heuring, "Digital fiber optic delay line memory," in *Digital Optical Computing II*, vol. 1215 (Los Angeles, CA), Jan. 1990. SPIE.

[11] Gilbert Strang, *Linear Algebra and Its Applications*, *2nd ed.* New York: Academic, 1980.

Donald M. Chiarulli received the B.S. degree in physics from Louisiana State University in 1976, the M.S. degree in computer science from Virginia Polytechnic Institute, Blacksburg, in 1979, and the Ph.D. degree in computer science from Louisiana State University in 1986.

He is an Assistant Professor of Computer Science at the University of Pittsburgh. His research interests include architectures and algorithms for the design and implementation of highly parallel computing systems, and optical technologies for interconnection networks and processing.

Dr. Chiarulli is a member of the IEEE Computer Society, SPIE, and the Optical Society of America.

Robert M. Ditmore, photograph and biography not available at the time of publication.

Steven P. Levitan (S'83-M'83) received the B.S. degree from Case Western Reserve University in 1972 and the M.S. and Ph.D. degrees both in computer science, from the University of Massachusetts, Amherst in 1979 and 1984, respectively.

He is the Wellington C. Carl Assistant Professor of Electrical Engineering at the University of Pittsburgh, Pittsburgh, PA. He was an Assistant Professor from 1984 to 1986 in the Electrical and Computer Engineering Department at the University of Massachusetts. In 1987 he joined the Electrical Engineering faculty at the University of Pittsburgh. His research interests include optical computing, parallel computer architecture, parallel algorithm design, and computer aided design for VLSI.

Dr. Levitan is a member of the IEEE Computer Society, ACM, SPIE, and OSA.

Rami G. Melhem received the B.E. degree in electrical engineering from Cairo University, Egypt, in 1976, the M.S. degree in mathematics computer science from the University of Pittsburgh in 1981, and the Ph.D. in computer science from the University of Pittsburgh in December 1983.

He is an Associate Professor of Computer Science at the University of Pittsburgh, Pittsburgh, PA. He has been an Assistant Professor of Computer Science at Purdue University from 1984 to 1986 and at the University of Pittsburgh from 1986 to 1989. His research interests include optical computing parallel systems, fault tolerant systems and the application of large computational arrays to scientific problems.

INTERNATIONAL JOURNAL OF OPTICAL COMPUTING, Vol. 0, 000–000 (1991)

OPTICAL MULTICASTING IN LINEAR ARRAYS

CHUNMING QIAO, RAMI G. MELHEM, DONALD M. CHIARULLI AND STEVEN P. LEVITAN

Department of Computer Science and Department of Electrical Engineering, University of Pittsburgh, Pittsburgh, PA 15260, U.S.A.

SUMMARY

In this paper, we use coincident pulse techniques to implement multicasting among processors connected by optical buses. First, we discuss two basic models of a unary addressing implementation. To reduce addressing latency and overcome system size limits, we propose a two-level addressing implementation in which multicasting introduces the problem of possibly addressing unintended processors (called *shadows*). We show how additional addressing pulses can be used to reduce these *shadows*. For regular multicasting patterns such as those often found in image processing and scientific applications, a shadow-free partition of the group to be multicasted can be systematically constructed. For arbitrary multicasting patterns, a simple, incremental partitioning algorithm is introduced. In summary, the two-level addressing implementation results in higher efficiency, lower minimum optical path requirements and potentially large speed-ups over the unary addressing.

KEYWORDS Multicasting Coincident pulse addressing Optical waveguides Shadow-free partition

1. INTRODUCTION

Coincident pulse techniques are based on two properties of optical pulse transmission, namely unidirectional propagation and predictable propagation delay per unit length. The technique was first introduced in the context of parallel memory addressing but was also applied to multiprocessor interconnection structures.[3,9,14] In this paper, coincident pulse techniques will be applied as an addressing mechanism for multicasting among optical bus connected processors.

In Section 2, we first review coincident pulse techniques as addressing mechanisms in two models of optical bus connected multiprocessor systems. We then show how multicastings can be implemented using unary addressing. In Section 3, two-level addressing is proposed to reduce the addressing latency and overcome the system size limit imposed by the unary addressing. However, multicasting with two-level addressing introduces the problem of possibly addressing unintended processors (called *shadows*). Simulation results of shadow reduction using additional addressing pulses are given. In Section 4, we show how shadows can be avoided by partitioning the group to be multicasted into shadow-free (SF) subgroups. We also show that speed-ups over the unary addressing can be achieved when multicasting using two-level addressing. Finally, we conclude the paper in Section 5.

† A preliminary short version of this paper appears in the proceedings of 1991 International Conference on Parallel Processing

1047-8507/91/000000-00$05.00

Received 25 January 1991
Revised 29 May 1991

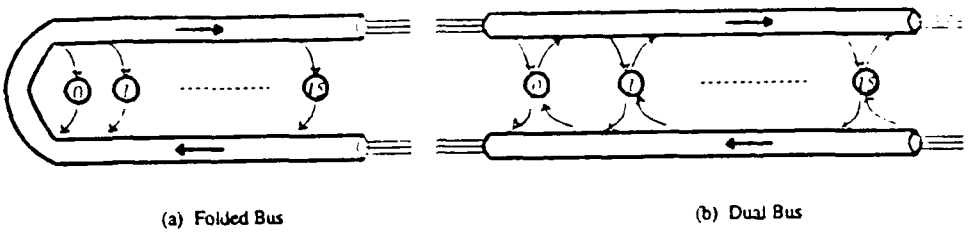


Figure 1. Two basic models

2. COINCIDENT PULSE ADDRESSING

As an introduction to using coincident pulse techniques as an addressing mechanism, we discuss two models of multiprocessor systems in which processors are connected by optical buses.[11] In the first model, called the *folded bus* model (see Figure 1(a)), each processor transmits on the lower half-segment of a bus, while receiving from the upper half-segment. In the second model, called the *dual bus* model (see Figure 1(b)), each processor is connected to two buses, one for downstream transmitting and upstream receiving and the other for upstream transmitting and downstream receiving.

An optical bus consists of three waveguides, one for carrying messages, one for carrying *reference pulses* and one for carrying *select pulses*, which we call the *message waveguide*, the *reference waveguide* and the *select waveguide* respectively. Messages are organized as *message frames*, which have a certain fixed length. The propagation delay on the reference waveguide is the same as that on the message waveguide but not the same as that on the select waveguide. A fixed amount of additional delay, which we show as loops (see Figure 2), are inserted onto the reference waveguide and the message waveguide.

The basic idea of using coincident pulse techniques as an addressing mechanism is as follows. Addressing of a destination processor is done by the source processor which sends a reference pulse and a select pulse with appropriate delays, so that after these two pulses propagate through their corresponding waveguides, a coincidence of the two occurs at the desired destination. The source processor also sends a message frame which propagates synchronously with the reference pulse. Whenever a processor detects a coincidence of a reference pulse and a select pulse, it reads the message frame. In essence, the address of a destination processor is unary encoded by the source processor using the relative transmission time of a reference pulse and a select pulse.

More specifically, let w be the pulse duration in seconds, and let c_h be the velocity of light in these waveguides. Define a unit delay to be the spatial length of a single optical pulse, that is $w \times c_h$. Starting with the fact that all three waveguides have equal intrinsic

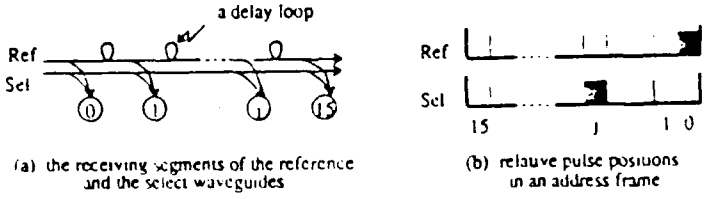


Figure 2. A unary addressing implementation

propagation delays, we add one unit delay (shown as one loop) on the reference waveguide and the message waveguide between any two adjacent receivers. In the folded bus model, this means that one unit delay is added between any two processors on the upper half-(receiving)-segment of the reference waveguide and the message waveguide as shown in Figure 2(a). Since there are no changes on the lower half-(transmitting)-segments of any waveguide and the message waveguide has exactly the same length as the reference waveguide, Figure 2(a) shows only the upper receiving segments of the select waveguide and the reference waveguide for a 16-processor system with a unary addressing implementation. Let T_{ref} be the time when processor i transmits its reference pulse and $T_{sel}(j)$ be the time when it transmits a select pulse. With delays added on the reference waveguide as in Figure 2(a), these two pulses will coincide at processor j if and only if

$$T_{sel}(j) = T_{ref} + j \tag{1}$$

where $0 \le i, j < N$ and N is the total number of processors in the system.

This means that for a given reference pulse transmitted at time t, the presence of a select pulse at time $t + j$ will address processor j while the absence of a select pulse at that time will not. Since we have $0 \le (T_{sel}(j) - T_{ref}) < N$, it is clear that N time units are needed to encode the complete address information of N processors with a unary addressing implementation. We define an *address frame* to be the address information in the form of a sequence of either the presence or the absence of select pulses relative to a given reference pulse. With a unary addressing implementation, an address frame has a length of N units long.

Figure 2(b) shows the position of the reference pulse and the select pulse addressing processor j at the transmission time of an address frame in the folded bus model. The relative position of the select pulse to the reference pulse will remain the same from the time the address frame is transmitted to the time it finishes propagation through the transmitting segment of the bus. However, the relative position will be changed as the address frame propagates through the receiving segment of the bus.

From the value of the term $(T_{sel}(j) - T_{ref})$, it is also clear that the relative positions of the reference pulse and select pulses are independent of the sending processor at the transmission time in this model. However, in the dual bus model, where a sending processor could transmit downstream and upstream using two buses, the necessary and sufficient condition for a select pulse to coincide with the reference pulse is

$$T_{sel}(j) = T_{ref} + j - i, \qquad \text{if } j \ge i \tag{2a}$$

or

$$T_{sel}(j) = T_{ref} + i - j, \qquad \text{if } j < i \tag{2b}$$

Noting that these two models have equivalent functionalities and similar operations, we will concentrate our discussions on the first model, namely the folded bus model throughout the rest of this paper.

One advantage of using coincident pulse techniques as an addressing mechanism is its applicability to multicasting. Traditional addressing mechanisms for multicasting, such as separate-addressing, multi-destination addressing and source routing [9-15] have been mainly developed for point-to-point networks and are inefficient, especially in bus connected

systems. Most recent research work[10] exploits *tree forwarding*[7] on broadcast networks which constructs multicast trees and uses group identifiers when multicasting. It requires explicit group formations of communicating processors.

Using coincident pulse addressing however, the sender can multicast to an arbitrary group of processors by sending an address frame containing one or more select pulses which are properly positioned so that each of them coincides with the reference pulse at one of the multicasting destinations. Once a processor detects a coincidence, it picks up a copy of the multicasted message frame which is synchronous with the reference pulse.

3. TWO-LEVEL ADDRESSING

Using unary addressing, an address frame is N units long. There are two reasons why we want to reduce the length of address frames by using two-level addressing. One has to do with efficiency. Unary addressing could be very inefficient in a large multiprocessing system where the address frame is longer than the message frame. The other reason has to do with the physical limitation of optical path length between two adjacent processors. One way to ensure that the frames sent by one processor do not collide with other frames sent by other processors is to arbitrate the bus to allow exclusive access by one processor at a time, as in References 4 and 11. Another way is to pipeline the bus. That is, to synchronize all processors such that they will send messages at the beginning of each cycle. The propagation delays between two adjacent processors should be large enough to prevent frames from overlapping as in References 8 and 11. If unary addressing is used, it is necessary for the optical path between any two adjacent processors to have a length of at least $N \times w \times c_b$ to prevent overlapping of the address frames. Although the required minimum optical path length can be reduced by shortening the pulse width w, the address frame length, which is linear in the system size, becomes a limiting factor.

A two-level addressing implementation divides the whole system into logical clusters. Addressing of a single destination is accomplished by using one level of unary addressing to select a particular cluster and another level of unary addressing to select an individual processor within the selected cluster. Two trains of select pulses are used, one for each level of addressing and their pulse trains are sent in parallel. Therefore, the length of address frames can be reduced as neither the number of clusters nor the size of any cluster is larger than the system size.

Assume that $N = n^2$ processors are linearly connected. If every n consecutive processors constitute one logical cluster, two-level addressing in this linear system is logically equivalent to addressing a two-dimensional array. More specifically, we can view the linear system as the result of embedding an $n \times n$ array in row major fashion. Each row of processors of the array is embedded into n consecutive processors in the linear system. Hence, selecting a logical cluster is equivalent to selecting a row while selecting an individual processor within a cluster is equivalent to selecting a column processor within a row.

3.1. *Two-level addressing in a linear system*

As mentioned above, we will view a linear system with $N = n \times n$ processors as a result of embedding an $n \times n$ array in row major fashion and use terms such as 'row', 'column' and 'diagonal' logically. As a logical equivalent to two-level addressing, a two-dimensional

addressing implementation uses two select waveguides: one to select a row, and another to select a column. A pulse sent on the row select waveguide will coincide with the reference pulse at all the processors in a particular row, while a pulse sent on the column select waveguide will coincide with the reference pulse at all processors in a column. In other words, a row-select pulse causes a row select trace and a column select pulse causes a column select trace. Pulses sent on these two select waveguides are denoted by $W1$ and $W2$ respectively.

A coincidence is said to occur at a given processor only if all *three* pulses, namely a reference pulse, a $W1$ pulse and a $W2$ pulse, coincide with each other at that processor. Since unary addressing is used when selecting a row, each pulse in the pulse train of $W1$ corresponds to one row. Similarly, each pulse in the pulse train of $W2$ corresponds to one column. Therefore, sending a reference pulse and a pair of one $W1$ pulse and one $W2$ pulse causes a coincidence at a processor that is located at the intersection of the corresponding row and column. More specifically, we denote L_1^i to be the pulse of $W1$ selecting row i and denote L_2^j to be the pulse of $W2$ selecting column j. By sending these two pulses and the reference pulse, the processor at row i and column j, which is processor $i \times n + j$ in an $N = n^2$ linear structure, is addressed. Figure 3 shows a logical two-dimensional view of addressing processor 10 with these two select pulses when $i = j = 2$ and $N = 16$.

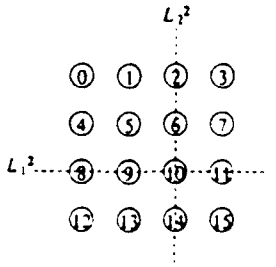
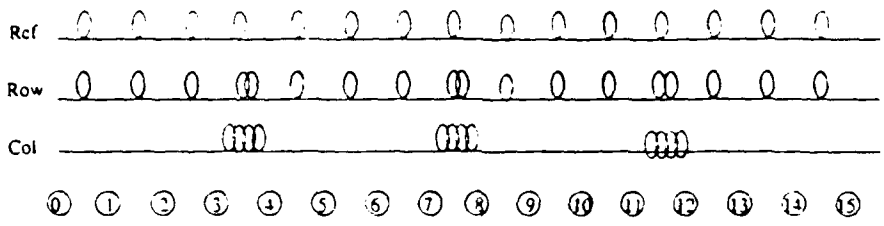


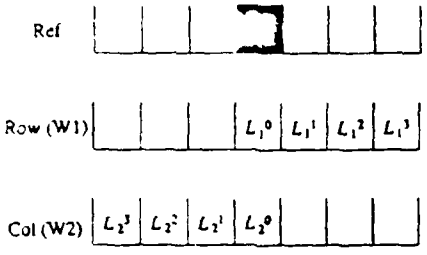
Figure 3. A logical view of addressing with two select pulses

In order to achieve the above coincidence pattern, we add, as in unary addressing, one unit delay between two processors on the receiving segments of the reference waveguide and the message waveguide. In addition, on the receiving segment of the row select waveguide ($W1$ waveguide), one unit delay is added between successive processors and an extra unit delay is added between two processors of successive rows. On the receiving segment of the column select waveguide ($W2$ waveguide), n unit delays are added between two receivers of successive rows. The amount of delay added on these two select waveguides can be obtained by solving a set of underconstrained equations (see Appendix). Again, because the message waveguide has exactly the same length as the reference waveguide and the lower half-(transmitting)-segments of all waveguides to not have any delays, only the receiving segments of the reference waveguide and the two select waveguides are shown in Figure 4(a). Taps from the waveguides to the processors are also omitted from the figure.

Let r_i and c_i be the row number and column number of processor i respectively. That is, $i = r_i \times n + c_i$ where $0 \le i < N$ and $0 \le r_i, c_i < n$. Let T_{ref} be the time when a processor transmits its reference pulse. And further assume a processor transmits a $W1$ pulse selecting row r_i at time $T_{ref}(L_1^{r_i})$ and a $W2$ pulse selecting column c_i at time $T_{ref}(L_2^{c_i})$. Given the



(a) the receiving segments of the reference, the row select and the column select waveguides



(b) Relauve pulse positions of $W1$ and $W2$ in an address frame

Figure 4. A two-level addressing implementation

added delays on three waveguides as shown in Figure 4(a), a coincidence of these three pulses will occur at processor j if and only if

$$T_{sel}(L_1^{r_j}) + (j + r_j) = T_{ref} + j \tag{3a}$$

and

$$T_{sel}(L_2^{c_j}) + n \times r_j = T_{ref} + j \tag{3b}$$

That is,

$$T_{sel}(L_1^{r_j}) = T_{ref} - r_j \tag{4a}$$

and

$$T_{sel}(L_2^{c_j}) = T_{ref} + c_j \tag{4b}$$

Since $0 \le r_j < n$, a $W1$ pulse is ahead of a reference pulse by 0 up to $n - 1$ units. The presence of a $W1$ pulse r_j units ahead of a reference pulse selects processors at row r_j while the absence does not. Similarly, a $W2$ pulse is 0 up to $n - 1$ units beyond a reference pulse and the presence of a $W2$ pulse c_j units beyond reference pulse selects processors at column c_j at each row while the absence does not. An address frame in the two-level addressing

implementation contains a train of $W1$ pulses and a train of $W2$ pulses and has a length of $2 \times n - 1$ units long. Figure 4(b) shows relative positions of the reference pulse and two trains of select pulses in an address frame at the time of transmission. Again, owing to the fact that an address frame will remain the same as it propagates through the transmitting segment of the bus in the folded bus model, the relative positionings of the reference pulse and both $W1$ and $W2$ pulses in an address frame at the time of transmission are independent of the sending processor.

Multicasting to a group of processors can be accomplished by sending a reference pulse, one train of $W1$ pulses with one or more pulses present and one train of $W2$ pulses with one or more pulses present along with a message frame. However, by doing so, coincidences may also occur at unintended processors, which we call *shadows*.[9] For example, when both processor i and j are addressed, the $W1$ train consists of two pulses, one for row r_i and another for row r_j. Similarly, the $W2$ train also consists of two pulses, one for column c_i and another for column c_j. In addition to processor i and j, the processor at row r_i and column c_j also detects a coincidence and picks up a copy of the multicasted message. So does the processor at row r_j and column c_i. Figure 5 shows a logical two-dimensional view of shadows at processor 1 and 10 as a result of multicasting to processors 2 and 9 in a 16 processor system.

3.2. *Shadow reduction*

As can be seen from the above example, shadows are created because of the unintended couplings of a $W1$ pulse with a $W2$ pulse. One way to reduce shadows is to further identify the intended pairs by using additional select waveguides, called *check* waveguides for carrying select pulses called *check* pulses. Check pulses are arranged such that they do not coincide with the reference pulse at places where shadows were created. Only processors at which coincidences of a reference pulse and all select pulses occur are addressed. This technique for shadow reduction was introduced in Reference 9 for addressing a two-dimensional memory structure. In the remainder of this section, we will show how to apply this technique to two-level addressing in a linear system. Note that having an additional check waveguide in two-level addressing is different from having three-level addressing. The latter would be logically equivalent to addressing a three-dimensional array. That is, addressing a single destination would require three select pulses. The address frame length would be further reduced while more shadows would be likely when multicasting with three-level addressing.

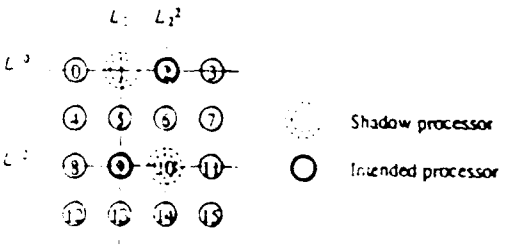
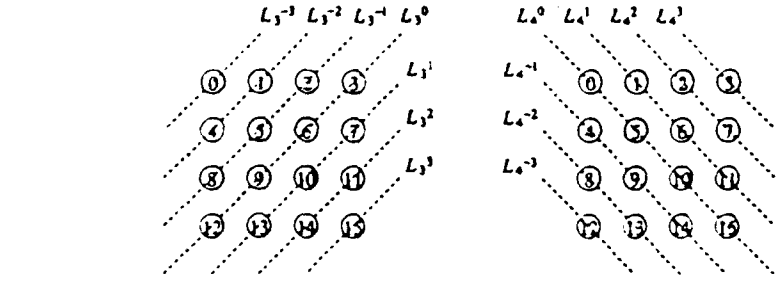
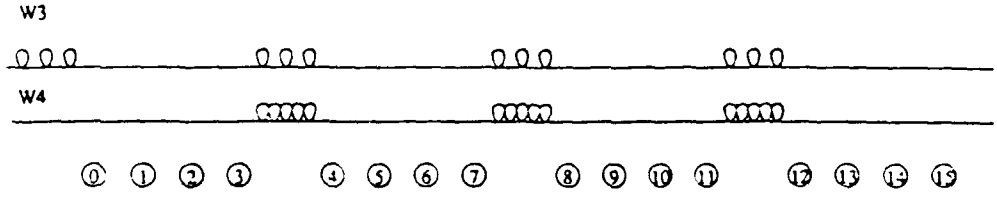


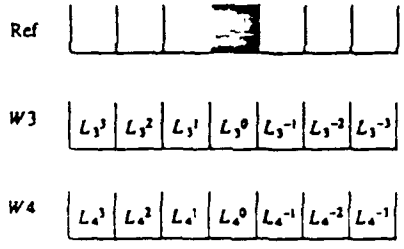
Figure 5. A logical view of shadows created at processor 1 and 10 as a result of multicasting to processor 2 and 9



(a) a logical view of traces of $W3$ pulses (b) a logical view of traces of $W4$ pulses



(c) the receiving segments of the $W3$ and the $W4$ waveguides



(d) relative pulse positions of $W3$ and $W4$ in an address frame

Figure 6. Adding two check pulses $W3$ and $W4$

One such set of check pulses are 45° diagonal select pulses, which will be called $W3$ hereafter. Another are $-45°$ diagonal select pulses, which will be called $W4$. Each pulse in a $W3$ or $W4$ train coincides with the reference pulse at all the processors that are on a 45° or $-45°$ diagonal line respectively, and therefore selects all processors on that diagonal line. Let L_3^{-k} be the $W3$ pulse selecting a 45° diagonal line which is k lines below or above the main 45° diagonal line respectively. Figure 6(a) shows a logical two-dimensional view of traces of $W3$ pulses. Similarly, let L_4^{-k} be the $W4$ pulse selecting a $-45°$ diagonal line which is k lines above or below the main $-45°$ diagonal line respectively. Figure 6(b) shows a logical two-dimensional view of traces of $W4$ pulses.

Again, the amount of delay that should be added on the $W3$ and $W4$ waveguides can be obtained by solving a set of underconstrained equations. Figure 6(c) shows only the receiving segments of both $W3$ and $W4$ waveguides with added delays. As a result, a $W3$

pulse and a $W4$ pulse will coincide with the reference pulse at the desired corresponding locations if and only if.

$$T_{sel}(L_3^{i,k}) = T_{ref} \pm k \tag{5a}$$

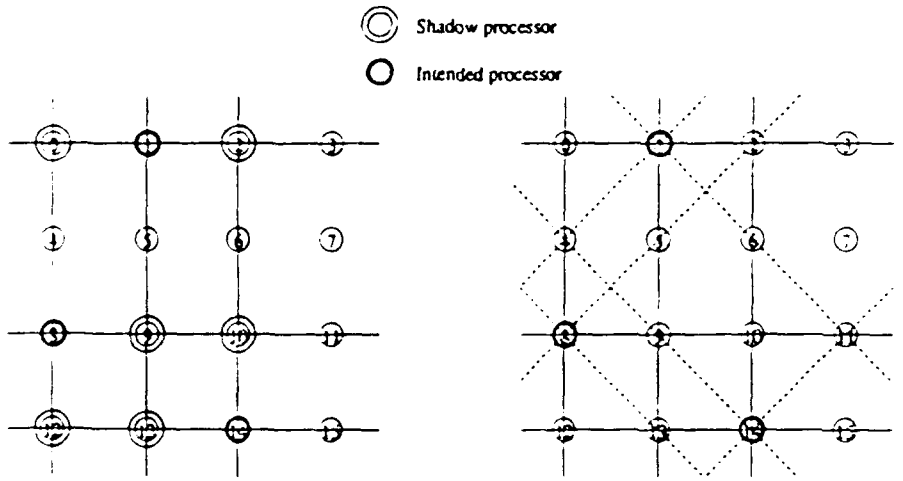
and

$$T_{sel}(L_4^{i,k}) = T_{ref} \pm k \tag{5b}$$

These two equations can be derived as follows. First, any processor at a 45° diagonal line which is k lines below the main diagonal line has the index number of $k \times n + i \times (n-1)$, for $1 \le i < (n-k)$. This means that the reference pulse will go through $k \times n + i \times (n-1)$ unit delays to arrive at the processor. Given that there are $n-1$ added unit delays at the beginning of each row on the receiving segment of the $W3$ waveguide as in Figure 6(c) (with $n = 4$), the $W3$ pulse L_3^k will coincide with the reference pulse at the processor if and only if $T_{sel}(L_3^k) + k + i) \times (n-1) = T_{ref} + n \times k + i \times (n-1)$, that is, $T_{sel}(L_3^k) = T_{ref} + k$. Similarly, we can completely derive the above equations.

In addition to a $W1$ train and a $W2$ train, an address frame now also contains a $W3$ train as well as a $W4$ train. Adding a $W3$ train and a $W4$ train does not change the length of an address frame, nor it does change the content of the $W1$ train or the $W2$ train. Figure 6(d) shows the relative positions of two trains of $W3$ and $W4$ pulses at the time of transmission of an address frame.

As an example, the two shadows in Figure 5 can be eliminated by using two $W3$ pulses, namely pulse $L_3^{4,-1}$ and pulse $L_3^{4,0}$. A more complicated example is shown in Figure 7. Figure 7(a) shows a logical view of six shadows created at processors 0, 2, 9, 10, 12 and 13 when multicasting to three processors 1, 8 and 14 without using any check pulses. Figure 7(b)



(a) a logical view of six shadows created without using any check pulses

(b) a logical view of eliminating shadows in (a) with three $W3$ pulses and three $W4$ pulses

Figure 7 Shadow reduction with check pulses

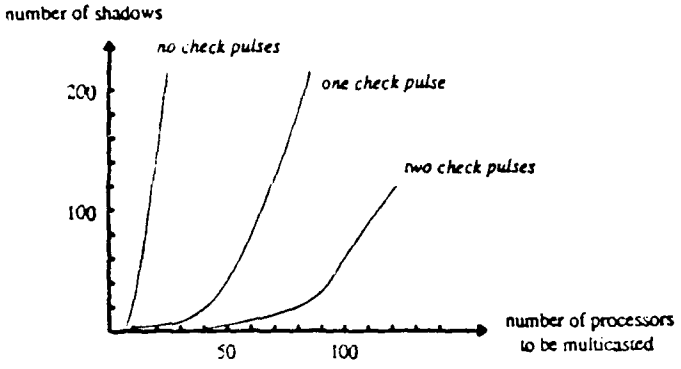


Figure 8. Shadows in a 512 × 512 array

shows a logical view of eliminating these shadows by adding three $W3$ pulses and three $W4$ pulses.

Figure 8 shows simulation results on the numbers of shadows created with different number of check pulses used. It is clear that the addition of check pulses cannot introduce new shadows. It can only reduce the number of existing shadows. However, adding a fixed number of check pulses cannot always completely eliminate shadows, since the theorem given in Reference 9 for a two-dimensional parallel memory structure also holds here.

4. SHADOW AVOIDANCE

Having established the relationship between a particular two-level addressing structure and its logically equivalent two-dimensional addressing representation, we will adapt to the usual notion of two-dimensional addressing in an $n \times n$ array in the following discussions for the purpose of simplicity. However, it is worth noting that techniques developed will be applied in physically linear systems with two-level addressing.

One way to avoid shadows when multicasting to a group of processors is to partition the whole group into several subgroups such that each subgroup is multicasted within one cycle without creating any shadows. More formally, assume a group of m processors is a set of linearly ordered processors denoted by $G = \{P_1, P_2, \ldots, P_m\}$. That is, for all $1 \le i \le m$, $0 \le P_i < N$ and $P_{i-1} < P_i$. Define a *shadow free* (SF) partition of the set G to be a number of subgroups S_1 through S_g such that for all $1 \le i, j \le g$ the following conditions are satisfied. (a) $S_i \cap S_j = \phi$ if $i \ne j$, (b) $\bigcup_{i=1}^{g} S_i = G$ and (c) each S_i can be multicasted in one cycle without any shadows.

A number of subgroups is called a *maximal* SF partition if it is a SF partition and if multicasting to more than one of the subgroups within one cycle will create a shadow. Therefore the number of subgroups of a maximal partition is the number of cycles needed to complete the multicasting to the whole group G.

Let processor X, where $0 \le X < N$, be a shadow created when multicasting to a group G in one cycle, clearly $X \notin G$. Define four shadow conditions (SC_i, $i = 1, 2, 3$ and 4) as follows.

SC_1: there is at least one processor in G that is in the same row as X
SC_2: there is at least one processor in G that is in the same column as X
SC_3: there is at least one processor in G that is in the same 45° diagonal line as X
SC_4: there is at least one processor in G that is in the same −45° diagonal line as X.

It can be verified that each of the conditions SC_i is necessary for X to be a shadow if the corresponding select pulse $W i$ is used to multicast the group G. The logical AND of all the necessary conditions becomes the sufficient condition. For example, if two pulses, $W 1$ and $W 2$, are used, then both condition SC_1 and SC_2 are the necessary conditions for a shadow to occur. The logical AND of the two is the sufficient condition.

4.1. *Regular multicasting patterns*

In some applications, such as finite element analyses and image processing, multicasting patterns can be quite regular. For example, a convolution of an $n \times n$ array involves multicastings of an element to its $w \times w$ neighbours, where w is the current window size. A group to be multicasted could also be all processors of a row, or of a column or of a diagonal line. By embedding a physical 2-D structure into our linear structure in the row-major fashion, these regular 2-D patterns can be characterized by a group of four parameters. More formally, in an embedded $n \times n$ system, we consider a group G of m processors starting with the processor numbered as k (called offset) with increment of d (called stride). Using the general notation in the beginning of the section, we have $G = \{k, k + d, \ldots, k + (m - 1) \times d\}$, where $0 \le k \le k + (m - 1) \times d < N = n^2$. We can use $G(k, d, m, n)$ to uniquely represent such a regular group. We call a group a *dense* group if d is less than n, a *sparse* group otherwise.

While we can make tradeoffs between the number of select waveguides used and the number of cycles needed to multicast to a group of processors, we will first analyse simple cases in which only two select waveguides are used. The results will be extended to cases in which four select waveguides are used.

Definition 1. A row of processors in a logical two-dimensional array is *incomplete* with regard to a group $G(k, d, m, n)$ if and only if the row contains two processors i and j such that $i \in G$, $j \notin G$ and $|j - i| = a \times d$ for some integer $a > 0$. A row is *complete* if and only if the row contains at least one processor of the group G and is not an *incomplete* one.

Definition 2. Define $I(k, d, m, n)$ to be the number of *incomplete* rows with regard to the group G.

Let the first processor of the group be $k = r_k \times n + c_k$ and the last processor of the group $l = k + (m - 1) \times d = r_l \times n + c_l$ for some integers $0 \le r_k, c_k, r_l, c_l < n$. And let condition 1 be that $c_k > d$, condition 2 be that $n - c_l > d$ and condition 3 be that $r_k \ne r_l$. There will be two *incomplete* rows, namely row r_k and row r_l if and only if all three conditions are true. There will be no *incomplete* rows if and only if neither condition 1 nor condition 2 is true. Otherwise, there will be only one *incomplete* row. Therefore, I has an upper bound of 2. Noting that for a *sparse* group $G(k, d, m, n)$ where $d \ge n$, neither condition 1 nor condition 2 is true, therefore $I = 0$.

† LCM stands for least common multiplier and GCD stands for greatest common divider.

Lemma 1. Two processors of a group $G(k, d, m, n)$ numbered as i and j, $i < j$, will be in the same column if and only if $j - i = b \times \text{LCM}(n, d)$ for some integer $b > 0$.

Proof. Let $i = r_i \times n + c_i$, and $j = r_j \times n + c_j$ as before. On the one hand, if $c_j = c_i$, then $j - i = (r_j - r_i) \times n$ and $r_j > r_i$. Since both processors i and j are in the same group, $j - i$ must be a multiplier of d, therefore $j - i$ should be a common multiplier of both n and d.

On the other hand, if $j - i = w \times \text{LCM}(d, n)$ for some integer $w > 0$, then clearly, $j - i$ is a multiple of n, which means processor i and j are at the same column. ■

Lemma 2. If there is a processor i in a group $G(k, d, m, n)$, $i = r_i \times n + c_i$, then processor j at the same column, $j = r_j \times n + c_i$, is also in the group G if

$$|r_j - r_i| = \frac{b \times d}{\text{GCD}(n, d)}$$

and row r_j is a *complete* one.

Proof. (By contradiction.) According to the definition of *complete* row, there must be a processor j at row r_j and $j \in G$. Clearly, $|j - i|$ should be a multiple of d. In addition, since processor j and i are at the same column and are $(b \times d)/[\text{GCD}(n, d)]$ rows apart, $|j - i| = |r_j - r_i| \times n$. That is,

$$|j - i| = b \times \frac{d \times n}{\text{GCD}(n, d)} = b \times \text{LCM}(n, d)$$

which is also a multiple of d. Therefore, $|j - j|$ must be a multiple of d also. If $j \notin G$, then row r_j is not a *complete* one, which contradicts the condition stated above. Therefore, $j \in G$. ■

According to the above lemmas, for a given group G, if we draw one vertical line at each processor of the group G, then all processors at the intersections of these vertical lines with *complete* rows which are $(b \times d)/[\text{GCD}(n, d)]$ apart will belong to the group G, and therefore can be multicasted without any shadows using row select pulses $W1$ and column select pulses $W2$.

Theorem 1. For a *dense* group $G(k, d, m, n)$ where $d < n$, the number of subgroups of a maximal partition with select pulses $W1$ and $W2$ has an upper bound of $d/[\text{GCD}(n, d)] + 1$.

Proof. We will prove the theorem by constructing a SF partition of the group.

First, we use one subgroup for processors of the group G at each *incomplete* row. According to the definition, there are l such subgroups and no shadows will occur in any of these subgroups because of the shadow condition SC_2. We partition the rest of group at remaining *complete* rows as follows.

Let $R = d/[\text{GCD}(n, d)]$. Starting at the first *complete* row, we put processors of the group at every R rows apart into a subgroup, creating exactly R subgroups. It can be similarly proved as in the Lemma 2 that no shadows will occur in any of these R subgroups. Therefore, a maximal SF partition will have at most $d/[\text{GCD}(n, d)] - 1$ subgroups. ■

It can be shown that the SF partition constructed in the above theorem is indeed a *maximal* SF partition. We can similarly prove the following theorem for a *sparse* group. First of all, I is equal to zero when $d \geq n$. Secondly, between any two rows which are R rows apart, where $R = d/[\text{GCD}(n, d)]$, there could be at most $(R \times n)/d$ processors belonging to the group and hence at most $[\text{LCM}(n, d)]/d$ *complete* rows. Therefore, when putting processors of the group every R rows apart into one subgroup, there are at most $[\text{LCM}(n, d)]/d$ subgroups in such a SF partition of the group G. This is stated in Theorem 2.

Theorem 2. For a *sparse* group $G(k, d, m, n)$ where $d \geq n$, the number of subgroups of a maximal partition with select pulses $W1$ and $W2$, has an upper bound of $[\text{LCM}(n, d)]/d$. ■

We can also prove the following theorems when using either one of the two diagonal select pulses, namely $W3$ or $W4$, with $W1$ instead of using $W2$ with $W1$ as in the above theorems.

Theorem 3. For a *dense* group $G(k, d, m, n)$ where $d < n$, the number of subgroups of a maximal partition with select pulses $W1$ and either $W3$ or $W4$, has an upper bound of $d/[\text{GCD}(n - 1, d)] + 1$ or $d/[\text{GCD}(n + 1, d)] + 1$ respectively. ■

Theorem 4. For a *sparse* group $G(k, d, m, n)$ where $d \geq n$, the number of subgroups of a maximal partition with select pulses $W1$ and either $W3$ or $W4$ has an upper bound of $[\text{LCM}(n - 1, d)]/d + 1$ or $[\text{LCM}(n + 1, d)]/d$ respectively. ■

The idea used to construct SF partitions in Theorem 3 and 4 is similar to the one used in Theorem 1 and 2. Processors of a group at certain number of rows apart will be put into one subgroup. These processors are at the intersections of these rows with either 45° diagonal lines or −45° diagonal lines depending on which one of the select pulses, $W3$ or $W4$, is used.

Since the additions of select pulses will not create any new shadows, we can choose a SF partition which has the *least* number of subgroups when all four select pulses discussed above are used.

Theorem 5. For a *dense* group $G(k, d, m, n)$ where $d < n$, the number of subgroups of a maximal partition with four select pulses $W1$, $W2$, $W3$ and $W4$ has an upper bound of

$$\frac{d}{\max(\text{GCD}((n-1), d), \text{GCD}(n, d), \text{GCD}((n+1), d))} + 1 \quad \blacksquare$$

Theorem 6. For a *sparse* group $G(k, d, m, n)$ where $d \geq n$, the number of subgroups of a maximal partition with four select pulses $W1$, $W2$, $W3$ and $W4$ has an upper bound of

$$\min\left(\frac{\text{LCM}((n-1), d)}{d} + 1, \frac{\text{LCM}(n, d)}{d}, \frac{\text{LCM}((n+1), d)}{d}\right) \quad \blacksquare$$

By applying Theorem 5 or Theorem 6 to some special instances of a group $G(k, d, m, n)$, such as a group of processors at one row with $d = 1$, a group of processors at one column with $d = n$, a group of processors at either diagonal lines with $d = n - 1$ or

$d = n + 1$, we know that each multicasting operation to such groups can be done in only one cycle without any shadows. These multicasting patterns are often seen in matrix manipulations, among many other applications.

The two theorems above can also be extended to a group of processors located in a small area of an $n \times n$ array. We delimit the area by an $\tilde{n} \times \tilde{n}$ array with $\tilde{n} \leq n$. The processors of the $\tilde{n} \times \tilde{n}$ array can be renumbered in a row major fashion from 0 to $\tilde{n}^2 - 1$. If a group of processors can be represented as $G(k, d, m, \tilde{n})$, we can partition the group similarly to what we did before. Since no shadow is possible outside the $\tilde{n} \times \tilde{n}$ area, by replacing every occurrence of n with $\tilde{n}$, the two theroems Theorem 5 and Theorem 6 can be applied to a group $G(k, d, m, \tilde{n})$ when $d < \tilde{n}$ or $d \geq \tilde{n}$ respectively.

The importance of this extension is that some of the most frequently used multicasting patterns in image processing[13] can now be analysed in term of SF partitions. For example, a four-neighbour group around any processor can be represented by a group with $k = 1$, $d = 2$, $m = 4$ and $\tilde{n} = 3$, or $G(1, 2, 3, 4)$, and each multicasting operation to the group can be done in one cycle. If we allow a processor to send a multicasting message to itself, then multicasting to its eight neighbours and itself, which is a group of $G(0, 1, 9, 3)$, can be done in one cycle. Similarly, multicasting to neighbouring $w \times w$ processors, as mentioned at the beginning of this section, can also be done in one cycle. By mapping hierarchical multigrids[2] or pyramid structures properly onto the logical 2-D structure, each processor can multicast to neighbouring processors or processors at the next level in one cycle.

4.2. *Arbitrary multicasting patterns*

As discussed above, there is a systematic way to construct a SF partition for any regular group. In order to construct a *maximal* SF partition, we need to merge subgroups of a SF partition together as long as the newly merged subgroups can still be multicasted without shadows. Similar partitioning procedures can also be used for arbitrary multicasting patterns. The proposed partitioning algorithm presented below consists of two parts. The first part is to construct a SF partition and the second part is to merge subgroups to construct a *maximal* SF partition.

For purposes of simplicity, we assume that three select pulses are used, namely row select pulses $W1$ and two diagonal select pulses $W3$ and $W4$. The first part of the algorithm partitions the whole group into at most $\lceil \frac{n}{2} \rceil$ non-empty subgroups. This is done by putting processors of the group at $\lceil \frac{n}{2} \rceil$ rows apart into one subgroup. Such a partition is a SF partition as stated in the following lemma.

Lemma 3. No shadows are possible when multicasting to a group of processors located at two rows which are at least $\lceil \frac{n}{2} \rceil$ apart in an $n \times n$ array with three select pulses $W1$, $W3$ and $W4$.

Proof. (By contradiction.) Let the two rows be r_1 and r_2. According to the condition SC_1 above, a shadow X must be at one of the two rows, assume it is row r_1. According to the shadow conditions SC_3 and SC_4 above, there must be two processors at row r_2 such that their respective 45° and −45° diagonal lines intersect at X. Therefore, the distance between these two processors should be two times the distance between the two rows r_1 and r_2, that is $2 \times \lceil \frac{n}{2} \rceil$, which is no less than n. Since these two processors are at the same row r_2, their distance could never exceed $n - 1$, hence no shadows are possible. ■

The second part of the algorithm first computes a forbidden set for each subgroup of the above SF partition. A row is said to be forbidden by a subgroup S_i if multicasting to S_i and processors of the group G located at that row will create at least one shadow. A forbidden set for a subgroup S_i is a set of rows forbidden by S_i. Two subgroups are merged together if neither contains processors at a row which is forbidden by the other. The new forbidden set for the merged subgroup is computed by adding certain rows into the union of the two original forbidden sets. When no further merges are possible, the algorithm stops and a maximal SF partition is constructed.

The time complexity of the algorithm is $O(n + m^2)$ where m is the number of processors in a group. This is because the first part of the algorithm takes $O(n)$ time. Careful analysis shows that the second part of the algorithm takes $O(m^2)$ time.

It is clear that no shadow is possible in a group of less than three processors when three select pulses are used. Therefore, any *maximal* SF partition will always have less than $\lceil \frac{m}{2} \rceil$ subgroups. Hence the algorithm will generate at most $\min(\lceil \frac{n}{2} \rceil, \lceil \frac{m}{2} \rceil)$ subgroups.

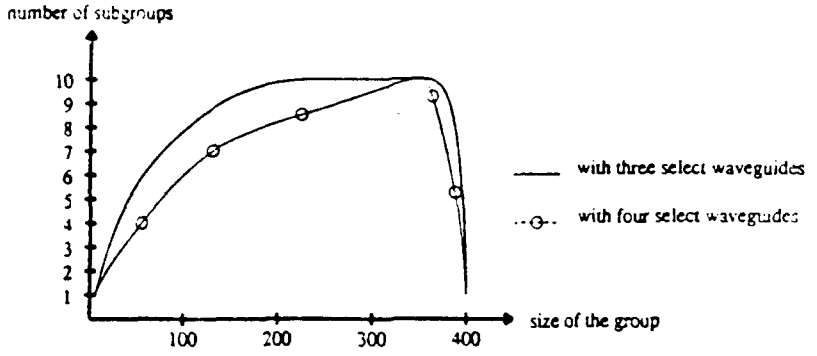
The partitioning algorithm can be executed incrementally when the multicasting pattern changes. Adding or deleting a processor from a group to be multicasted involves possibly spliting an existing subgroup, forming a new subgroup which contains processors at the same row and finally re-merging any subgroups which have since been changed.

If the column select pulses $W2$ are used instead of the row select pulses $W1$, the algorithm above can be adapted accordingly by partitioning columns which are $\lceil \frac{m}{2} \rceil$ columns apart into subgroups and merging them into a *maximal* SF partition. If all four select pulses mentioned above are used, we can start with either SF partitions and augment the condition which determines if a row (or a column) is forbidden by a certain subgroup and merge subgroups together to achieve a *maximal* SF partition.

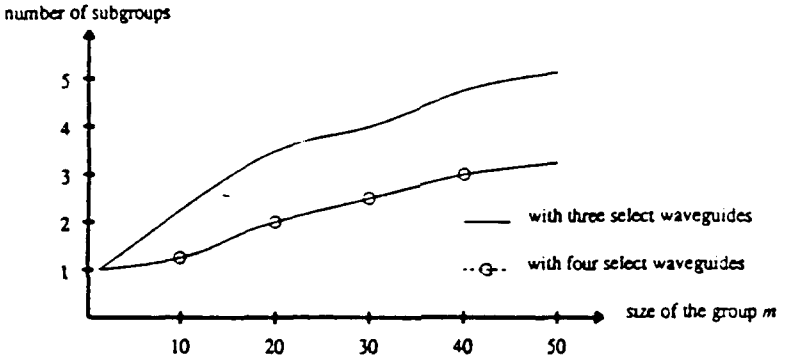
Figure 9 shows the simulation results on the number of subgroups generated by the algorithm. For a 20×20 processor system in Figure 9(a), when all 400 processors are multicasted, there are no unintended processors at all and that is why the number of subgroups is reduced to 1. If the number of subgroups in a maximal SF partition of a group G is g, then the time needed to transmit g address frames, one in each cycle, is $g \times (2 \times n - 1) \times c_h$ in the two-level addressing implementation. However, $N \times c_h$ is needed in a unary addressing implementation. Therefore, the speed-up is at least $n/(2 \times g)$. Noting that g can not exceed $n/2$, the worst-case speed-up is 1. As shown in Figure 9(b), with three select pulses, the average number of subgroups needed to multicast to 50 processors in a 50×50 processor system is only about 5. Thus, a speed-up of 5 is obtained.

5. CONCLUSION

In this paper, coincident pulse techniques have been applied as an efficient addressing mechanism for multicasting among multiprocessors connected by optical buses. Two basic models of a unary addressing implementation have been discussed, and a two-level addressing implementation has been proposed to reduce the address frame length. Two approaches to deal with the shadow problem have been presented. One approach reduces the number of shadows by using check pulses. Another approach avoids possible shadows by constructing SF partitions. It has been shown that for regular multicasting patterns, SF partitions can be constructed systematically and processors can multicast to their communicating processors within one cycle in many applications. A partitioning algorithm has also been presented for arbitrary multicasting patterns. The overall results of the two level addressing implementation are higher efficiency, lower minimum optical path requirements and potential speed-ups.



(a) number of subgroups vs. size of the group in a 20×20 system



(b) number of subgroups for a group of $m = n$ processors in a $n \times n$ system

Figure 9. Simulation results of the partitioning algorithm

This reinforces our belief that coincident pulse techniques are a promising addressing mechanism which can be applied in both parallel memory structures and multiprocessor systems. Finally, we note that in this paper, many technical aspects of pulse generation, coincidence detection, power distribution and other related issues have not been discussed. They can be found in References 5, 9 and 12.

APPENDIX

As mentioned in Section 3.1, the amount of delay that is added on the row select and the column select waveguides can be obtained by solving a set of underconstrained equations. In the following discussion, a folded bus model of $N = n^2$ processor is assumed and some of the previous defined notation is used. Additional notation is defined as follows.

Row (i), *Col* (i):
the number of delay loops added on the receiving segment of the row select and the column select waveguides, respectively, between processor $i - 1$ and processor i, where $0 \le i < N$.

$D(L_1^i)$, $D(L_2^j)$:
the transmission time of the row select pulse L_1^i and the column select pulse L_2^j, respectively, relative to the transmission time of the reference pulse, where $0 \le i < n$.

Given that there is one delay loop between any two adjacent processors on the receiving segment of the reference waveguide, we have the following set of equations for the row select waveguide:

$$T_{sel}(L_1^i) + \sum_{j=0}^{k} Row(j) \begin{cases} = T_{ref} + k, & \text{if } i \times n \le k < (i+1)n \\ \neq T_{ref} + k, & \text{otherwise} \end{cases} \quad \text{(A.1a)}$$

Note that, the number of delay loops added on the reference waveguide has been fixed therefore a degree of freedom has been removed already. Equation (A.1a) states that the pulse L_1^i should coincide with the reference pulse at all processors at row i but should not coincide with the reference pulse at any other processors. Since $T_{sel}(L_1^i) - T_{ref} = D(L_1^i)$, we can simplify the above equation to

$$D(L_1^i) + \sum_{j=0}^{k} Row(j) \begin{cases} = k, & \text{if } i \times n \le k < (i+1)n \\ \neq k, & \text{otherwise} \end{cases} \quad \text{(A.1b)}$$

Clearly, two different pulses cannot be transmitted at the same time, therefore, the following equation has to be satisfied also:

$$(D(L_1^i) \neq D(L_1^j), \quad \text{if } i \neq j \quad \text{(A.1c)}$$

These two equations, namely (A.1a) and (A.1b), are underconstrained since both $D(L_1^i)$s and $Row(j)$s are variables. Note that the values of $D(L_1^i)$s will determine the address frame length and therefore we choose to fix them first and solve the equation for $Row(j)$s. If $D(L_1^i)$s are fixed such that $D(L_1^i) = i$, we can solve the above equations to get

$$Row(j) = \begin{cases} 0, & \text{if } j = i \times n \\ 1, & \text{otherwise} \end{cases}$$

Note that, if the $D(L_1^i)$s are fixed such that $D(L_1^i) = -i$, we will get the same result as in Section 3.

Similarly, we can have the following set of equations for column select waveguide:

$$D(L_2^j) + \sum_{i=0}^{k} Col(i) \begin{cases} = k, & \text{if } k = i - mn, \text{ where } 0 \le m < n \\ \neq k, & \text{otherwise} \end{cases} \quad \text{(A.2a)}$$

$$D(L_2^i) \neq D(L_2^j), \quad \text{if } i \neq j \quad \text{(A.2b)}$$

It can be shown that by fixing $D(L_2^j) = i$, we can get the result as in Section 3. Equations and their solutions for check waveguides can be similarly constructed.

ACKNOWLEDGEMENTS

This research is supported in part by a grant from the Air Force Office of Scientific Research under contract # AFOSR-89-0469 and in part by a grant from the National Science Foundation under contract MIP-90-03553.

REFERENCES

1. L. Aguilar, 'Data gram routing for Internet multicasting', *ACM Sigcomm 84 Computer Communications Review*, 14, 58–63 (1984).
2. T. Chan and R. Schreiber, 'Parallel networks for multi-grid algorithms: architecture and complexity', *SIAM Journal on Scientific and Statistical Computing*, 6, 698–711 (1985).
3. D. Chiarulli, R. Melhem and S. Levitan, 'Using coincident optical pulses for parallel memory addressing', *IEEE Computer*, 20, 48–58 (1987).
4. D. Chiarulli, S. Levitan and R. Melhem, 'Optical bus control for distributed multiprocessors', *Journal of Parallel and Distributed Computing*, 10, 45–54 (1990)
5. D. Chiarulli, S. Levitan and R. Melhem, 'Demonstration of an all-optical addressing circuit', *Technical Digest 1991 OSA Topical Meeting on Optical Computing*, Salt Lake City, 1990.
6. Y. Dalal and R. Metcalfe, 'Reverse path forwarding of broadcast packets', *Communications of ACM*, 21, 1040–1048 (1978).
7. A. Frank, L. Wittie and A. Bernstein, 'Multicast communication on network computers', *IEEE Software*, 2, 49–61 (1985).
8. Z. Guo, R. Melhem, R. Hall, D. Chiarulli and S. Levitan, 'Array processors with pipelined optical busses', *Proceedings of the 3rd Symposium on the Frontiers of Massively Parallel Computations* (also to appear in the *Journal of Parallel and Distributed Computing*, 1991), 1990, pp. 353–342.
9. S. Levitan, D. Chiarulli and R. Melhem, 'Coincident pulse techniques for multiprocessor interconnection structures', *Applied Optics*, 29, 2024–2039 (1990).
10. P. McKinley and J. Liu, 'Multicast tree construction in bus based networks', *Communications of ACM*, 33, 29–42 (1990).
11. R. Melhem, D. Chiarulli and S. Levitan, 'Space multiplexing of waveguides in optically interconnected multiprocessor systems', *The Computer Journal*, 32, 362–369 (1989).
12. M. Nassehi, F. Tobagi and M. Marhic, 'Fiber optic configurations for local area networks', *IEEE Journal on Selected Areas in Communication*, SAC-3, 941–949 (1985).
13. A. Netravali and J. Limb, 'Picture coding: a review', *Proc IEEE*, 68, 336–406 (1980)
14. C. Qiao and R. Melhem, 'Time-division optical communications in multiprocessor array', Tech. Rep 91-14, Department of Computer Science, University of Pittsburgh (March 1991)
15. D. Wall, 'Selective broadcast in packet-switched networks', *Proc. Sixth Berkeley Workshop Distributed Data Management and Computer Networks*, 1982, pp. 239–258.

JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING 12, 269–282 (1991)

Pipelined Communications in Optically Interconnected Arrays*

ZICHENG GUO, RAMI G. MELHEM, RICHARD W. HALL, DONALD M. CHIARULLI, AND STEVEN P. LEVITAN

Departments of Electrical Engineering and Computer Science, University of Pittsburgh, Pittsburgh, Pennsylvania 15261

Two synchronous multiprocessor architectures based on pipelined optical bus interconnections are presented. The first is a linear pipeline with enhanced control strategies which make optimal use of the available communication bandwidth of the optical bus. The second is a two-dimensional architecture in which processors are placed in a square grid and interconnected to one another through horizontal and vertical pipelined optical buses. These architectures allow any two processors to communicate with each other using one (for the linear case) or two (for the two-dimensional case) pipelined bus cycles. Further, they permit all processors to have simultaneous access to the buses using slots within a pipelined cycle. We show that the architectures have simple control structures and that well-known processor interconnections, e.g., the complete binary trees and the hypercube networks, can be efficiently embedded in them. These architectures have an effectively higher bandwidth than conventional bus configurations and appear to be good candidates for a new generation of hybrid optical–electronic parallel computers.

1. INTRODUCTION

Two-dimensional meshes of processors have been extensively studied in various forms and augmentations [23, 26, 37]. Large-scale implementations of two-dimensional meshes have been built [2, 10, 17]. However, since the communication diameter of an $n \times n$ mesh is $O(n)$, different approaches have been considered to augment the communication capabilities of the mesh to reduce this diameter. Meshes have been augmented with global buses [3, 10, 11, 35], reducing the communication diameter but giving only very small bandwidth improvements. Row and column bus augmentations [29, 30] have yielded both a low communication diameter and adequate bandwidth for certain classes of algorithms. Interconnection networks have been considered for augmenting rows and columns in a mesh including trees [27, 28, 39] and compounded graphs [18, 19]. The binary hypercube can also be viewed in this context as a two-dimensional mesh with horizontal and vertical hypercube interconnections [18, 19].

* This work was, in part, supported by Air Force Grant AFOSR-89-0469 and by NSF Grant MIP-8901053.

One of the simplest mesh augmentation schemes is the row and column bus augmentation. However, exclusive write access to buses is a major contributor to the low bandwidth of bus interconnections. A unique property of optics provides an alternative to this exclusive access, namely, the ability in optics to pipeline the transmission of signals through a channel. In electronic buses, signals propagate in both directions from the source, while optical channels are inherently directional and have precise predictable path delays per unit distance. Hence, a pipeline of optical signals may be created by the synchronized directional coupling of each signal at specified locations along the channel. This property has been used to parallelize access to shared memory [5], to enhance the bandwidth in bus-connected multiprocessor systems [22], and to minimize the control overhead in networking environments [38].

In this paper, we present two multiprocessor architectures, called *Array Processors* with *Pipelined Buses* (APPB), which employ optical bus interconnections in processor arrays. In Section 2 we review the basic principle of pipelining messages on optical buses. In Section 3 we introduce our linear APPB, where processors are connected with a single optical bus. We present efficient approaches to message routing and network embedding for the linear APPB as well as techniques for enhancing the bus utilization through enhanced control functions. In Section 4 we introduce our two-dimensional APPB, where processors are interconnected with horizontal and vertical optical buses. We discuss routing and embedding issues for this new architecture. We show how binary tree and hypercube interconnections can be effectively embedded and identify key design issues for effective embeddings of arbitrary interconnections. In Section 5 we compare the efficiency of the pipelined bus communication model with that of nonpipelined buses and of store and forward communications in nearest-neighbor structures. Finally, Section 6 contains concluding remarks.

2. MESSAGE PIPELINING ON OPTICAL BUSES

Consider the system of Fig. 1a, where n processors, each having a constant number of registers, are connected through a single optical waveguide (bus). Each processor is coupled to the optical waveguide with two passive couplers, one for injecting (writing) signals on the waveguide and the other

0743-7315/91 $3.00

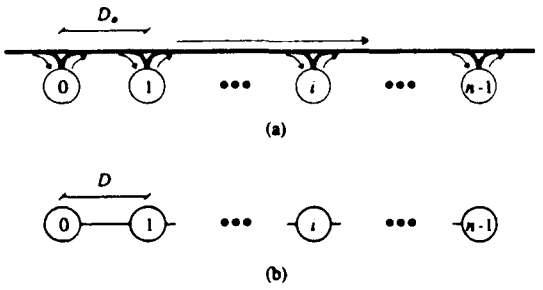


FIG. 1. (a) A system of n processors connected with a single optical waveguide (bus). (b) A linear array of n processors with nearest-neighbor connections.

for receiving (reading) signals from the waveguide [20, 40]. Each receiving coupler passively taps a percentage (typically 5–10%, depending on the coupling ratio) of the optical signal power available on the bus. Thus the couplers do not introduce any delay to the propagation of optical signals along the bus. However, the degradation of signal power does place an upper limit on the number of processors that can be connected on the bus [8]. As in the case of electronic buses, each processor j communicates with any other processor i by sending a message to i through the common bus. However, because optical signals propagate in one direction, a processor j may send signals to another processor i only if $i > j$.

Assume that a message on an optical bus consists of a sequence of optical pulses, each having a width w in seconds. The existence of an optical signal of width w represents a binary bit 1, and the absence of such a signal represents a 0. Note that w includes a time for electro-optical conversions, rise and fall times, and propagation delay in the latch of the receiver circuits [6]. For analytical convenience, we let D_o be the optical distance between each pair of adjacent nodes (it will become clear that the distance between two adjacent nodes need not be equal) and τ be the time taken for an optical pulse to traverse the optical distance D_o. To transfer a message from a node j to node i, $i > j$, the sender j writes its message on the bus. After a time $(i - j)\tau$ the message will arrive at the receiver i, which then reads the message from the bus.

The properties of unidirectional propagation and predictable path delays of optical signals may be used advantageously. Specifically, unlike the electronic case, where the writing access to the bus by each node must be mutually *exclusive*, all nodes in the system of Fig. 1a can write on the bus *simultaneously*, provided that the following collision-free condition [22] is satisfied.

$$D_o > bwc_e. \tag{1}$$

where b is the number of binary bits in each message, and c_e is the velocity of light in the waveguide. Clearly if this condition is satisfied and the system is synchronized such that every node starts writing a message on the bus at the same instant, then no two messages injected on the bus by any two distinct nodes will collide. Here by colliding we mean that two optical signals injected on the bus by any two distinct nodes arrive at some point on the bus simultaneously. This kind of synchronized pulse generation is restrictive but it can be met in several ways [21]. An optically distributed clock can be broadcast without skew to each node, or electro-optical switches can be used in place of sources to "switch in" pulses generated from a single source. With this condition satisfied, every node can, in parallel, send a message to some other node, and the messages will all travel from left to right on the bus in a pipelined fashion, as shown in Fig. 2. Thus we use the term *pipelined bus*. In the rest of this paper we always assume that the collision-free condition (1) is satisfied.

To facilitate our discussion in subsequent sections we define some terms. Let τ be defined as before and n be the number of nodes on the pipelined optical bus. We define $n\tau$ as a *bus cycle* and correspondingly τ as a *petit cycle*. Note that a bus cycle is the time taken for an optical signal to traverse the entire length of the optical bus. For the discussion in this section, we do not include in a bus cycle the time taken to prepare and process a message before it can be injected on the bus. This time is explicitly introduced in our performance analysis in Section 5. If every node is writing a message simultaneously on the bus, then each node has to wait for at least a bus cycle to inject its next message. Note that each cycle on the pipelined bus may be emulated by n cycles in a linear array with nearest-neighbor communications shown Fig. 1b. Comparison of the two interconnection schemes is made in Section 5.

Let us look at a simple routing task where each node transmits a message and each node is programmed to receive a message from the kth node (if it exists) to its left. All nodes start injecting messages at the beginning of a bus cycle, and all the messages travel on the optical bus in pipelined fashion without collision. By waiting for a specific interval of time, a node can selectively read the message intended for it as that message passes by the node. In our example, each node i is to receive a message from node $i - k$ and thus must read its message from the bus after $k\tau$ time from the beginning of the bus cycle. In this way, a message routing pattern in which each node sends a message to the kth node to its right has been realized. In fact, as will be seen, we can realize various message routing patterns in a simple, straightforward way.

3. LINEAR ARRAY PROCESSORS WITH PIPELINED BUSES

In the system of Fig. 1a, messages can be transmitted only from left to right. To allow message passing from right to

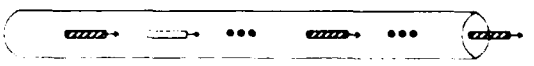


FIG. 2. Message pipelining on the optical bus. A blank rectangle indicates "no signal," implying that some processor is not sending a message

left, another optical bus is used, as shown in Fig. 3a. In this figure, we have two optical buses; the upper one is used for sending messages from left to right, and the lower one is used for sending messages from right to left. Each node can write and read messages on either bus as desired. Obviously signals in different buses do not disturb one another; that is, the two buses can support two separate pipelines. The system in Fig. 3a is our architecture of linear APPB. For convenience the linear APPB in Fig. 3a is schematically drawn as in Fig. 3b.

To specify the time at which a node should receive a message, we introduce a control function $twait(i)$, which is defined as the time that node i should wait, relative to the beginning of the bus cycle, before reading the message sent to it from some other node j. Thus

$$twait(i) = (i - j)\tau.$$

If τ is considered as a time unit, then $twait$ can be interpreted in terms of the number of such time units and thus be written $twait(i) = i - j$. Clearly if $twait(i) > 0$, then the message is to be received from the left; if $twait(i) < 0$, then the message is to be received from the right. If $twait(i) = 0$, then no message should be received by node i. The value of $twait(i)$ can be stored in a *wait register*, and more than one such register may be used if a node is to receive more than one message in one bus cycle.

This $twait$ control function, however, has the disadvantages that it depends crucially on timing accuracy and is sensitive to the optical distance D_0 between two adjacent nodes. An equivalent control function, $mwait$, that does not have these disadvantages may be defined if we require that each node inject a message, real or dummy, every bus cycle. In this case we define $mwait(i)$ as the number of messages that node i should skip before reading its message. For example, if $mwait(i) = \gamma$, then node i should receive the $|\gamma|$th message that passes i on the bus. That is, it has to wait until $|\gamma| - 1$ messages have passed and then it reads its own message. The sign of γ determines on which bus the message should be received. Clearly $mwait$ is equivalent to $twait$ and either control function may be used. For convenience we simply write the control function as $wait$, and we assume that the optical distance between each pair of adjacent nodes i and $i + 1$ is constant.

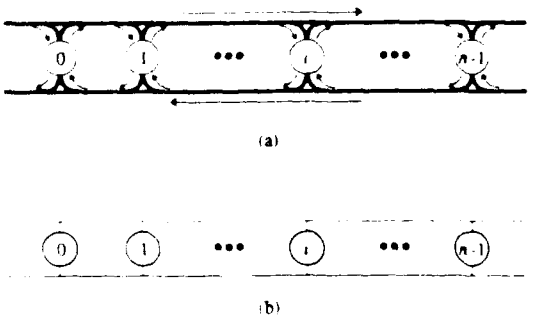


FIG. 3. (a) Linear array processors with pipelined buses (APPB). (b) A schematic drawing of (a).

The control function $wait$ can only be used when the communication pattern is known to the receiver in the sense that the receiver knows from which node the message is to be received. In cases where the communication pattern is unknown to the receiver, the coincident pulse techniques [5, 21] may be used such that an addressing pulse and a reference pulse coincide at the detector of the receiver, thereby addressing it. In this paper we use $wait$ for addressing since the communication patterns which we discuss are known to the receiver.

In the following we present techniques for message routing and network embedding in the linear APPB. For the purpose of evaluating the communication efficiency, we note that a lower bound on the number of bus cycles needed to transfer H messages in the linear APPB is $\lceil H/n \rceil$, where n is the number of nodes on the optical bus. This lower bound is obtained by assuming a perfectly even distribution of messages along the bus at each bus cycle, that is, *every* node has one message to send at *each* bus cycle.

3.1. *Message Routing in Linear APPB*

Various message routing patterns can be realized in a simple, straightforward way. Since a routing pattern is determined by the $wait$ functions, we need only determine these $wait$ functions for each routing pattern. The most common patterns are:

One-to-One. The system executes a $SEND(j, i)$ instruction, which means that a message is to be transferred from node j to node i. Thus, $wait(i) = i - j$, where i is a *single* specific node.

Broadcast. The system executes $BROADCAST(j)$, which means that node j broadcasts a message, and all other nodes i will receive that message. In this case, $wait(i) = i - j$ for *all* $i \neq j$.

Semigroup Communication [4]. The system executes a $SEMIGROUP(i)$ instruction, which says that some global information, e.g., extrema and sum, is to be computed and stored at node i. This task can be accomplished by having the linear APPB logically function as a tree with the root being node i. Later in this section we present embeddings of binary trees which facilitate such a tree emulation task.

Permutations. For each node j to send a message to a node $i = PERM(j)$, where $PERM(\ )$ is an arbitrary permutation, we set $wait(i) = i - j$ for *all* i.

We see that the computation of $wait(i)$ is very simple and uniform. The only difference among the $wait$ functions for different message routing patterns is that the nodes involved

are different. It is clear that all these communication tasks can be performed using a single bus cycle, except the semigroup communication, which takes $\log(n)$ bus cycles. Note that, in the linear APPB, message passing between two nonneighboring nodes is nearly as efficient as that between two neighbors. Specifically, a message takes τ more time to pass one more node on the optical bus. This is not the case in the linear array with nearest-neighbor connections shown in Fig. 1b, where to pass a node, en route to another node, a message has to go through a router. In this sense we may say that the APPB is communication efficient, and in particular global-communication efficient.

3.2. *Embedding Binary Tree and Hypercube Networks into Linear APPB*

In this subsection we show how to embed other interconnection networks into the linear APPB. Our first example is the embedding of complete binary tree networks. To show that a binary tree network can be embedded in the linear APPB it is sufficient to find the *wait* function for each processor in the linear APPB such that the desired routing pattern is accomplished.

Let L be the number of levels of a complete binary tree and let the root of the tree be node 1. Each node i, $i \geq 1$, which is not a leaf node has two children, $2i + \delta$, where $\delta = 0, 1$, corresponding to i's left and right child, respectively (see Fig. 4a for an example). Consider an embedding in which node i in the tree is mapped to node $i - 1$ in the linear APPB. For convenience, we call this embedding E_{t1} (see Fig. 4b). In E_{t1}, the wait functions for node i to receive a message from its children are:

$$wait_{c,\delta}(i) = \begin{cases} i - (2i + \delta) = -(i + \delta), & i < 2^{L-1}, \\ 0, & \text{otherwise.} \end{cases}$$

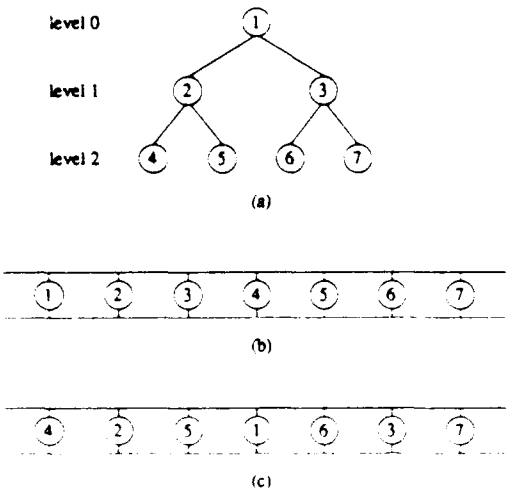


FIG. 4. Embeddings of complete binary trees in the linear APPB. (a) A binary tree. (b) The first embedding, E_{t1}. (c) The second embedding, E_{t2}.

Thus, to realize children-to-parent message routing each parent should wait for $wait_{c,0}(i)$ and $wait_{c,1}(i)$ time to read the messages from its left and right child, respectively. Clearly this routing task can be performed using one bus cycle.

For parent-to-children message transfer in E_{t1}, each parent has two messages to send to its two children, respectively. In this case, two bus cycles are needed to carry out such a routing task, one to send messages to left children and one to send messages to right children. Let $wait_{p,0}(j)$ and $wait_{p,1}(j)$ be the wait functions for a left child and right child, respectively, to receive a message from its parent. Then, during the first cycle we have

$$wait_{p,0}(j) = \begin{cases} j - \dfrac{j}{2} = \dfrac{j}{2}, & j = \text{even}, \\ 0, & \text{otherwise}, \end{cases}$$

and during the second cycle we have

$$wait_{p,1}(j) = \begin{cases} j - \dfrac{j-1}{2} = \dfrac{j+1}{2}, & j = \text{odd, and } j \neq 1, \\ 0, & \text{otherwise.} \end{cases}$$

Mapping each node i in the binary tree network onto node i (or $i - 1$ as was just done above) in the linear APPB is a straightforward approach. Using this straightforward approach we can embed any type of network in the linear APPB. This approach, however, may not give a good embedding in the sense that it may take more time than needed, in number of bus cycles, to accomplish a given communication task. As is seen next, another tree embedding, E_{t2}, has a better communication efficiency than E_{t1}.

Embedding E_{t2} may be viewed as pressing the binary tree from the root down until all the nodes fall in the level of the leaf nodes (see Fig. 4c). In this embedding the two children of a node i are on opposing sides of i. Thus the parent-to-children routing pattern, as well as the children-to-parent routing pattern, may be accomplished in one bus cycle. Specifically, if i is a node at level l, where l is the integer satisfying $2^l - 1 < i < 2^{l+1}$, then the wait functions for i to receive the messages from its two children are

$$wait_{c,\delta}(i) = \begin{cases} (-1)^{\delta} 2^{L-l-2}, & i < 2^{L-1}, \\ 0, & \text{otherwise.} \end{cases}$$

The parent-to-children message routing pattern in E_{t2} is different from that in E_{t1} in that the two messages from a parent will travel on two different buses. Then the two messages from each parent node can be simultaneously injected on the two buses, respectively, in the same bus cycle. Hence, the parent-to-children routing pattern can be accomplished in one bus cycle. $wait_{p,\delta}$ can be determined by noting that

$wait_{p,\delta}(j) = -wait_{c,\delta}(i)$, where i is the parent of j. That is, the wait functions for parent-to-children message transfer are

$$wait_{p,\delta}(j) = \begin{cases} (-1)^{\delta+1} 2^{L-l-1}, & j > 1, \\ 0, & j = 1. \end{cases}$$

Next, we consider a k-dimensional binary hypercube in which the nodes are numbered such that if nodes i and j are neighbors across dimension h, $1 \leq h \leq k$, then $|i - j| = 2^{h-1}$ (see Fig. 5a). Let E_{c1} be the embedding of this k-cube into a linear APPB such that each node i in the hypercube is mapped into node i in the linear APPB. With this embedding, a node in the hypercube may send a distinct message to each of its k neighbors if each node sends one message to one neighbor in each bus cycle. For example, at the hth bus cycle a message is sent from each node to its neighbor at distance 2^{h-1}. To accomplish this, the time that a node i has to wait during the hth bus cycle before receiving a message from its neighbor along the hth dimension is

$$wait_h(i) = \pm 2^{h-1}.$$

In our discussions so far, we have allowed each node to send only one message on each bus during each bus cycle. In other words after placing a message on the bus in the current cycle, all nodes must wait until the next cycle to initiate the next message. In the following subsection, we show that such a wait is not always necessary.

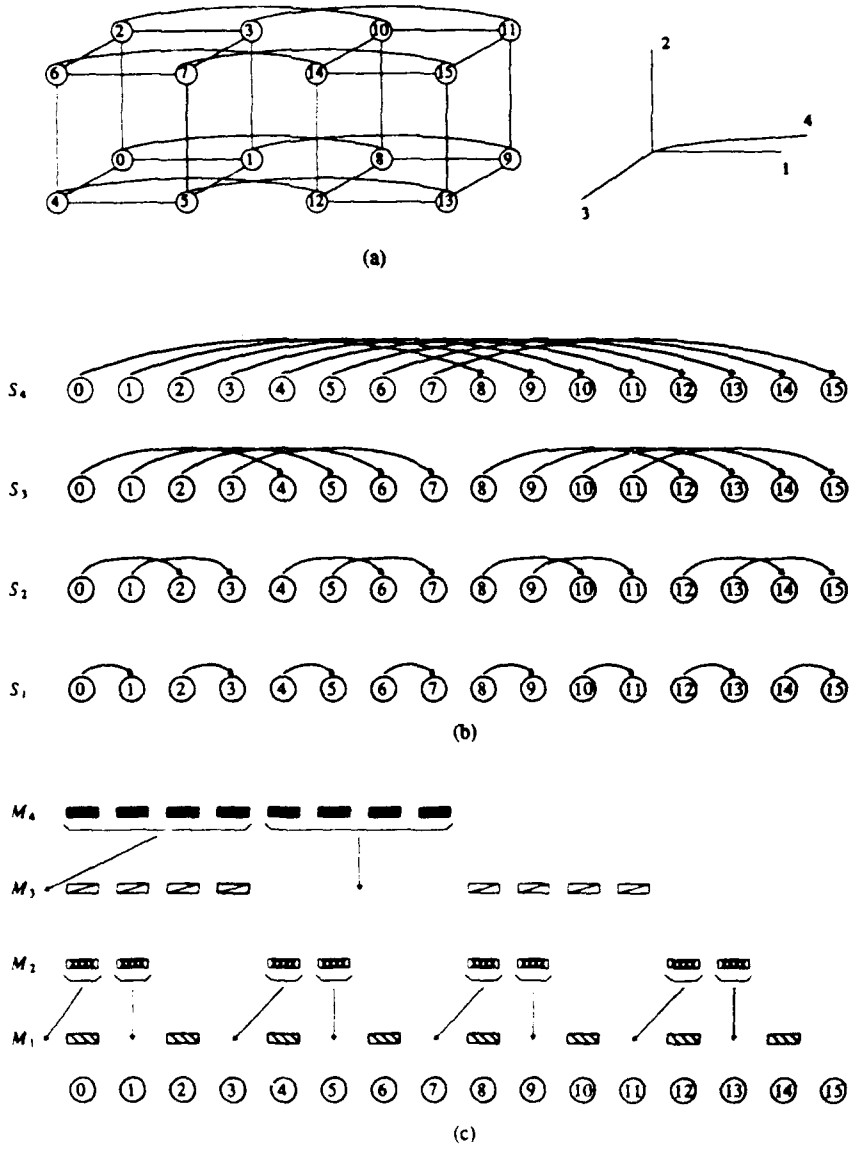


FIG. 5. (a) A binary hypercube and its dimension assignment. (b) Message routing patterns in the hypercube. (c) Message distribution in the hypercube.

3.3. *Interleaved and Overlapped Pipelining*

Up until now, we have required that each node send only one message on each bus in one bus cycle and that the transmission of messages be initiated at the beginning of a bus cycle. Given these two restrictions, no specific control function was needed for the initiation of messages. However, if some node does not have a message to send during a bus cycle, a slot of one petit cycle in duration will be created. Interleaved pipelining is a technique which tries to fully utilize the communication capacity of the pipelined bus by inserting a message into any available slot. This may be accomplished if a node is allowed to place more than one message on the same bus within a bus cycle, but at different petit cycles. To allow for this flexibility, a control function $send_g(j)$ must be used to specify the time, relative to the beginning of a bus cycle, at which node j should write its gth message on the bus.

To show how interleaved pipelining works, let us now examine the routing patterns in E_{c1}. Since message transfers in opposite directions on the two buses of the linear APPB form two separate and symmetric pipelines, we need to look at only one direction. Consider the left-to-right message transfer in E_{c1}, and define k sets, $S_h = \{j \mid 0 \leq j < n, 0 \leq (j \bmod 2^h) < 2^{h-1}\}$, $1 \leq h \leq k$, of nodes for the k-cube. That is, S_h is obtained by partitioning the n nodes of the hypercube into 2^h-node groups and including in S_h the first 2^{h-1} nodes in each group. For example, for the 4-cube in Fig. 5a, we have $S_1 = \{0, 2, 4, 6, 8, 10, 12, 14\}$, $S_2 = \{0, 1, 4, 5, 8, 9, 12, 13\}$, $S_3 = \{0, 1, 2, 3, 8, 9, 10, 11\}$, and $S_4 = \{0, 1, 2, 3, 4, 5, 6, 7\}$. Note that all the k sets, S_h, have the same cardinality 2^{k-1}, and each contains node 0. Hence, in the realization of the binary k-cube using a linear APPB, there are k routing patterns. In the hth pattern, $1 \leq h \leq k$, the nodes in set S_h send messages to their neighbors along the hth dimension in the hypercube, as indicated with the arrowed curves in Fig. 5b. Correspondingly, the messages can be divided into k sets, M_h, $1 \leq h \leq k$, which are sent by the k sets of nodes S_h, respectively. For the routing patterns in Fig. 5b, these message sets are shown in Fig. 5c.

Using interleaved pipelining, the messages in the two sets M_{2s-1} and M_{2s}, $1 \leq s \leq k/2$, are interleaved and sent in the same bus cycle. Let $send_1(j)$ and $send_2(j)$ be the times at which node j writes its messages in M_{2s-1} and M_{2s}, respectively, on the bus during bus cycle s. Correspondingly, let $wait_1(i)$ and $wait_2(i)$ be the wait functions for a node i to receive the messages in M_{2s-1} and M_{2s}, respectively, during bus cycle s. Then, for interleaved pipelining we have the following *send* functions for a node j at bus cycle s, $1 \leq s \leq k/2$:

$$send_1(j) = 0, \quad j \in S_{2s-1},$$

$$send_2(j) = \begin{cases} 0, & j \in S_{2s} \text{ and } 2^{2s-2} \leq (j \bmod 2^{2s}) < 2^{2s-1}, \\ 2^{2s-2}, & j \in S_{2s} \text{ and } 0 \leq (j \bmod 2^{2s}) < 2^{2s-2}. \end{cases}$$

The corresponding *wait* functions are

$$wait_1(i) = i - j, \quad j \in S_{2s-1},$$

$$wait_2(i) = \begin{cases} i - j, & j \in S_{2s} \text{ and } 2^{2s-2} \\ & \quad \leq (j \bmod 2^{2s}) < 2^{2s-1}, \\ i - j + 2^{2s-2}, & j \in S_{2s} \text{ and} \\ & \quad 0 \leq (j \bmod 2^{2s}) < 2^{2s-2}. \end{cases}$$

A node for which the *send* or *wait* function is not defined above should not send or receive any message. Note that the times determined by these *send* and *wait* functions are with respect to the beginning of each bus cycle s. Also note that since the receiving node i knows the id of the sending node j (since they are neighbors in the k-cube), it knows which of the two values of $wait_2(i)$ should be used. As an example, the interleaved pipelining for the messages in Fig. 5c is achieved by interleaving message sets M_1 and M_2 in the first bus cycle and M_3 and M_4 in the second bus cycle. The arrowed lines in Fig. 5c show how the messages are being interleaved, and the resulting message pipelines are shown in Fig. 6a.

It can be seen that using interleaved message pipelining, the total communication time taken for each node to send a message to each of its neighbors is $k/2 + 1$ bus cycles, where the last bus cycle is due to the time needed to clear out the first $n/4$ messages (sent by nodes 0 through $n/4 - 1$) in M_k that were inserted in front of M_{k-1}. Comparing with k bus cycles, the time needed if each node sends one message per bus cycle, our savings in the communication time is $(k - 2)/2$ bus cycles. Although this savings is significant there are still unused slots from the rightmost nodes on the bus, as can be seen from the message pipeline at time $t = 16$ in Fig. 6a. We next show how to utilize these empty slots using overlapped pipelining.

In overlapped pipelining, we pipeline the message pipelines obtained from interleaved pipelining by allowing the messages for bus cycle s to be initiated before bus cycle $s - 1$ terminates, as long as message collision does not occur. For this purpose we define a new control function, $step_s$, which specifies the time, with respect to the beginning of the first bus cycle, at which the messages for bus cycle s are initiated. Clearly, savings in communication time is possible if $step_s - step_{s-1} < n\tau$. In this case, we avoid confusion by calling the bus cycles *message transfer steps*.

In E_{c1}, the control function $step_s$, $1 \leq s \leq k/2$, specifies when S_{2s-1} and S_{2s} should start sending their messages. Specifically let $step_1 = 0$ and let $step_s$, $1 < s \leq k/2$, be the time interval in number of petit cycles between the initiations of steps 1 and s. Then, messages from step s and step $s - 1$ will not collide if

$$step_s = step_{s-1} + n - \frac{3}{4} 2^{2s-2}, \quad 1 < s \leq \frac{k}{2}.$$

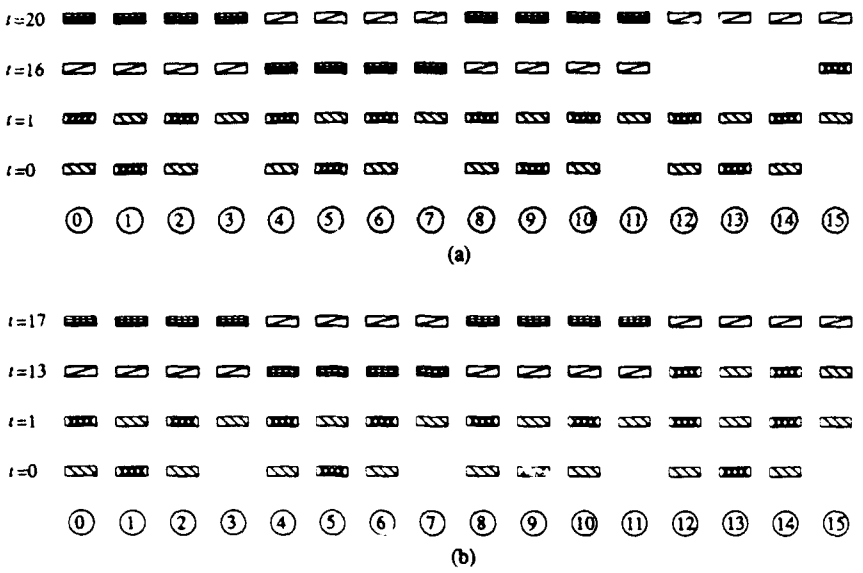


FIG. 6. (a) Interleaved pipelining and (b) overlapped pipelining of messages in the 4-cube. (t is measured in petit cycles.)

The *send* and *wait* functions defined in the previous subsection are still applicable here, but they are now defined with respect to the time determined by $step_s$, the beginning of transfer step s, rather than the beginning of each bus cycle s. Figure 6b shows the result of overlapped pipelining of the message pipelines in Fig. 6a. Note that in interleaved pipelining there was also some overlapping between the two message pipelines generated in two consecutive bus cycles, as can be seen from the message pipeline at time $t = 16$ in Fig. 6a. But, as has been mentioned previously, interleaved pipelining does not fully utilize the pipelined bus.

These control functions *step*, *send*, and *wait* together result in a minimized total communication time. To show this we first note that since the cardinality of M_h, $1 \leq h \leq k$, is $n/2$, the total number of messages is $kn/2$. Thus, if we assume that the message distribution over processors is perfectly even in each bus cycle (every processor has a message to send in each bus cycle), then the time needed for transferring these messages is at least $\lceil kn/2n \rceil = k/2$ bus cycles, or equivalently $kn/2$ petit cycles. In our case, however, such an assumption of even message distribution does not hold. For example, no message can be inserted on the bus at processor $n - 1$ in the first bus cycle, as can be seen from the message pipeline at time $t = 0$ in Fig. 6b. Now we compute the total time, in number of petit cycles, using the control functions determined above. It can be shown that

$$step_{k/2} = \sum_{s=2}^{k/2} \left(n - \frac{3}{4} 2^{2s-2} \right) = \left(\frac{k}{2} - \frac{5}{4} \right) n + 1.$$

The time due to $send_2$ at step $k/2$ is $2^{k-2} = n/4$. Finally it takes n petit cycles for the bus to clear out. Therefore the total time in number of petit cycles is

$$\left(\frac{k}{2} - \frac{5}{4} \right) n + 1 + \frac{n}{4} + n = \frac{k}{2} n + 1.$$

Finally, we note that interleaved message pipelining may also be applied to binary tree routing patterns. From our previous discussion we know that the parent-to-children message routing in E_{t1} has to be done in two bus cycles and that the same message routing task can be performed using a single bus cycle in E_{t2}. Communication efficiency in E_{t2} can be further improved by using interleaved message pipelining because during parent-to-children message transfer only every other node is sending a message. Thus each parent can send two messages to each child in one bus cycle.

4. TWO-DIMENSIONAL ARRAY PROCESSORS WITH PIPELINED BUSES

Linear optical buses have the disadvantage that message transfer may incur $O(N)$ time delay in an N-processor system. To reduce this delay to $O(\sqrt{N})$, we consider two-dimensional APPBs. In a two-dimensional APPB, each node is coupled to four buses as shown in Fig. 7a, where the two horizontal buses are used for passing messages horizontally in the same way as before, and the two vertical buses are used for passing messages vertically in a similar way. For convenience we diagram our two-dimensional APPB as in Fig. 7b. Each node in a two-dimensional APPB of size $N = m \times n$ will be given two identifications, one being a pair of numbers (x, y), $0 \leq x < m$, $0 \leq y < n$, indicating the row–column position of the node in the two-dimensional APPB, and the other being the row-major index, $i = xn + y$, $0 \leq i < N$, of the node. Corresponding to the bus cycle defined for the linear case, in the two-dimensional APPB we define $n\tau$ and $m\tau$ as a *row bus cycle* and a *column bus cycle*, respectively, where τ is a petit cycle as defined previously. When there is no confusion, e.g., while talking about message transmissions in a row, we simply say a *bus cycle* instead of a *row bus cycle*.

4.1. *Message Routing in Two-Dimensional APPB*

A unique issue that arises in the two-dimensional APPB is the relay of messages. Specifically, suppose a message is to be transferred from node (x_1, y_1) to node (x_2, y_2), with $x_1 \neq x_2$ and $y_1 \neq y_2$. Then the message may first be sent from (x_1, y_1) to (x_1, y_2), which is the node at the intersection of row x_1 and column y_2, in the first bus (a row bus cycle) and then from (x_1, y_2) to (x_2, y_2) in the second bus cycle (a column bus cycle). That is, the message has to be buffered at node (x_1, y_2) at the end of the first bus cycle and then relayed to its destination in the second bus cycle. For the purpose of relaying the message, we define a control function *relay* for node (x_1, y_2) as

$$relay[(x_1, y_2)] = y_2 - y_1,$$

which indicates that node (x_1, y_2) will read a message from a row bus at time $|y_2 - y_1|$ (relative to the start of the row bus cycle) and then write that message on the proper column bus at the beginning of the following column bus cycle. If $relay[(x_1, y_2)] = 0$, then no message is to be relayed by node (x_1, y_2). Clearly, in the worst case up to n messages have to be relayed and, therefore, n relay buffers are needed at the relaying node. Now we are ready to show how the four most commonly used message routing patterns discussed in the previous section can be realized in the two-dimensional APPB.

One-to-One. The system executes a $SEND[(x_1, y_1), (x_2, y_2)]$ instruction, which requires that node (x_1, y_1) send a message to node (x_2, y_2). We have $relay[(x_1, y_2)] = y_2 - y_1$ (in row bus cycle), and $wait[(x_2, y_2)] = x_2 - x_1$ (in column bus cycle). This communication takes two bus cycles.

Broadcast. The system executes a $BROADCAST[(x, y)]$ instruction, which states that node (x, y) broadcasts the same message to all other nodes (x_i, y_i). In a row bus cycle, (x, y) broadcasts the message to nodes (x, y_j), $y_j \neq y$. Then in the following column bus cycle all (x, y_j), including (x, y), broadcast the message in their corresponding columns. Thus $relay[(x, y_j)] = y_j - y$, and $wait[(x_i, y_j)] = x_i - x$. This communication also takes two bus cycles.

Semigroup Communication. This corresponds to the execution of $SEMIGROUP[(x, y)]$, which says that some global information is to be computed and stored at node (x, y). This task can be accomplished using two linear semigroup operations, one in rows and the other in a column. That is, first we view each row as a linear APPB and do $SEMIGROUP(y)$ in all rows. Then in column y, we perform $SEMIGROUP(x)$. Thus $2 \log(n)$ bus cycles are needed for this task.

Permutations. Let $PERM[(x, y)]$ be an arbitrary permutation. To avoid using n relays at each node, we can use a three-phase routing approach [24, 32] or equivalently a three-bus-cycle approach in the two-dimensional APPB. In this approach the first bus cycle is a "preprocessing" step which distributes messages in each row such that the messages going to the same row will occupy different columns. Then the second and third bus cycles will route the messages to their destination row and destination node, respectively. We note that for arbitrary permutations this approach implies the use of a centralized controller which would compute the message destinations for the preprocessing step. This calculation requires the construction of a bipartite graph and its partitioning into complete matchings, which would dominate the time complexity for the total task of computing and implementing an arbitrary permutation. In applications where a permutation can be precomputed, this time cost can be amortized over many subsequent applications of the permutation.

4.2. *Embedding Binary Trees in Two-Dimensional APPB*

As mentioned previously, arbitrary message routing and permutations in two-dimensional APPB may require n relaying buffers in each node in the worst case. In this subsection we present an embedding for a binary tree network in which only one relay buffer is needed to route messages. An embedding of an L-level complete binary tree into a two-dimensional APPB with $n = 2^k$ columns may be obtained by (i) mapping levels $0, \ldots, k-1$ of the tree to row 0 of the two-dimensional APPB and (ii) mapping level l, $k \leq l < L$, of the tree to the 2^{l-k} rows, $2^{l-k}, 2^{l-k}+1, \ldots, 2^{l-k+1} - 1$, of the APPB such that the two children of the same parent are mapped into two adjacent rows in the same column as the parent. Specifically we define our embedding of a binary tree network into the two-dimensional APPB by a mapping $F(i) = (F_x(i), F_y(i))$, which maps each node i, $1 \leq i < 2^L$, in the tree to a node $(F_x(i), F_y(i))$ in the two-dimensional APPB. Let i be a node at level l, $0 \leq l < L$, in the binary tree. The mapping is defined by

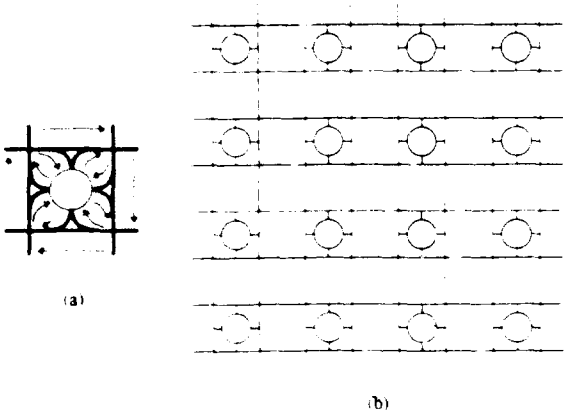


FIG. 7. Two-dimensional APPB. (a) A processor coupled to four waveguides in the two-dimensional APPB. (b) A schematic drawing of the two-dimensional APPB.

$$F_x(i) = \begin{cases} 0, & 1 \le i < 2^k, \\ 2^{l-k} + i \bmod 2^{l-k}, & 2^k \le i < 2^L, \end{cases}$$

and

$$F_y(i) = \begin{cases} i, & 1 \le i < 2^k, \\ \left\lfloor \dfrac{i \bmod 2^l}{2^{l-k}} \right\rfloor, & 2^k \le i < 2^L. \end{cases}$$

As an example the embedding for the 4-level binary tree in Fig. 8a is shown in Fig. 8b. Let us call this embedding E_{t3}. E_{t3} has the following properties: (i) Parent nodes i, $1 \le i < 2^{k-1}$, and their children are in row 0; (ii) parent nodes i, $2^{k-1} \le i < 2^k$, which are in row 0, have their children in row 1; and (iii) parent nodes i, $2^k \le i < 2^{L-1}$, and their children are in the same column. Properties (i) and (ii) are obvious. Here we prove only (iii). Since in the binary tree each parent node i has two children $2i + \delta$, $\delta = 0, 1$, to prove (iii) we need only show that $F_y(i) = F_y(2i + \delta)$ for $2^k \le i < 2^{L-1}$. For that, let i be a parent node at level l, where $k \le l < L - 1$ and $i = p2^l + q$ for some integers p and q such that $0 \le q < 2^l$. Then

$$\begin{aligned} F_y(2i + \delta) &= \left\lfloor \frac{(2(p2^l + q) + \delta) \bmod 2^{l+1}}{2^{l+1-k}} \right\rfloor \\ &= \left\lfloor \frac{(p2^{l+1} + 2q + \delta) \bmod 2^{l+1}}{2^{l+1-k}} \right\rfloor \\ &= \left\lfloor \frac{2q + \delta}{2^{l+1-k}} \right\rfloor = \left\lfloor \frac{q}{2^{l-k}} \right\rfloor = F_y(i). \end{aligned}$$

It is now clear that the relay function is not needed for message transfer between parent nodes i and their children if $1 \le i < 2^{k-1}$ or $2^k \le i < 2^{L-1}$. However, such a relay is needed if $2^{k-1} \le i < 2^k$. The wait and relay functions for E_{t3} are obtained in the following.

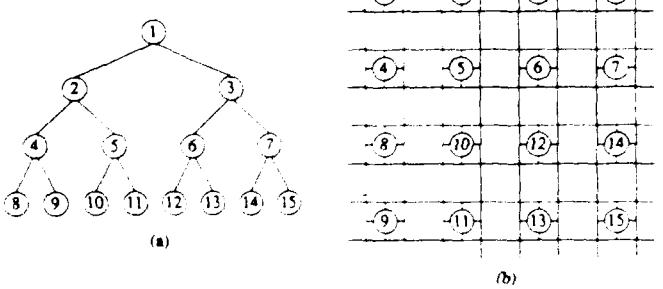


FIG. 8. (a) A 4-level binary tree. (b) Its embedding, E_{t3}, in the two-dimensional APPB.

Let $wait_{c,\delta}[(x, y)]$, where $(x, y) = F(i)$, be the *wait* functions for a parent node i to receive a message from its left and right child for $\delta = 0$ and 1, respectively. For the case $1 \le i < 2^{k-1}$, the results for the linear APPB directly give $wait_{c,\delta}[(x, y)] = -(y + \delta)$. For the case $2^k \le i < 2^{L-1}$, let i be at level l, $k \le l < L - 1$, and $i = p2^{l-k} + q$. Then

$$F_x(i) = 2^{l-k} + i \bmod 2^{l-k} = 2^{l-k} + q;$$

$$F_x(2i + \delta) = 2^{l+1-k} + (2i + \delta) \bmod 2^{l+1-k}$$

$$= 2^{l+1-k} + (p2^{l+1-k} + 2q + \delta) \bmod 2^{l+1-k}$$

$$= 2^{l+1-k} + 2q + \delta;\ wait_{c,\delta}[(x, y)] = F_x(i) - F_x(2i + \delta)$$

$$= (2^{l-k} + q) - (2^{l+1-k} + 2q + \delta) = -(x + \delta).$$

$wait_{p,\delta}(j)$ can be obtained by recalling that $wait_{p,\delta}(j) = -wait_{c,\delta}(i)$, where i is the parent of j.

For the case where $2^{k-1} \le i < 2^k$, a *wait* and a *relay* function are needed. Let $relay_{c,\delta}[(0, y)]$, $0 \le y < 2^k$, be the *relay* function of node $(0, y)$ for relaying the message from a child node (again, $\delta = 0$ for the left child and $\delta = 1$ for the right child) to its parent. Then we can show that

$$relay_{c,\delta}[(0, y)] = -1, \quad 0 \le y < 2^k,$$

$$wait_{c,\delta}[(0, y)] = 2^k - y - \delta, \quad 2^{k-1} \le y < 2^k.$$

Note that each node $(0, y)$ needs to relay only *one* child-to-parent message with the message from left (right) child being relayed by $(0, y)$ with y even (odd), and that even though node 0 is not a node in the tree, it helps relay messages. Also note that $relay_{c,\delta}$ is applicable to column bus cycles. Now let $relay_{p,\delta}[(0, y)]$, y even (odd), be the relay function for node $(0, y)$ to relay the message from a parent $(0, Y)$ to its left (right) child for $\delta = 0$ (1). Then $relay_{p,\delta}$ is easily obtained from $relay_{p,\delta}[(0, y)] = -wait_{c,\delta}[(0, Y)]$. And $wait_{p,\delta}$ is determined as in the linear case.

4.3. *Network Embeddings Requiring No Relays*

Embedding E_{t3} still requires one message relay for communication between two neighboring nodes in binary trees. To further improve the communication efficiency, in this subsection we show how to obtain embeddings of binary trees as well as hypercubes such that no such message relay is needed. Two approaches may be used to eliminate message relays by intermediate nodes: a hardware approach and a "software" approach. In the hardware approach, optical switches are used at the intersections of row and column buses to switch an optical signal, say, from a row bus to a column bus, without requiring relay by an intermediate processor [15]. In this paper we consider the "software" ap-

proach, which relies on designing embeddings such that all neighboring processors in a network are mapped into the same row or column in the two-dimensional APPB. Thus, no message relay is needed and no *relay* function is required. This improves the communication efficiency significantly. However, it has the disadvantage that nodes in the APPB may not be fully utilized.

A basic measure that is usually used to evaluate the quality of an embedding of a source graph $G_1 = \{V_1, U_1\}$ with a set of nodes V_1 and a set of edges U_1 into a mesh architecture with a set of nodes V_2 is the *expansion cost*, which is defined as the ratio of the number of nodes in the target mesh to the number of nodes in the embedded graph. Another measure useful for such evaluation is the *dilation cost*. Specifically, the dilation of an edge $u \in U_1$, which is mapped to a path Q in the target mesh, is $|Q| - 1$, where $|Q|$ is the number of nodes on Q. However, the mesh model corresponding to that of APPBs is different from those studied previously [1, 13, 34] because the efficiency of the communication between any two nodes in the same row or column in an APPB does not depend on the distance between these two nodes. Therefore the criterion that is to be satisfied by an embedding is different from previously studied criteria. Specifically, it is desirable to obtain an embedding in which any two neighboring nodes in the source graph are mapped into either the same row or the same column in the two-dimensional APPB, thus allowing them to communicate with each other using a single bus cycle. An embedding which satisfies this requirement will be said to satisfy the *alignment condition*. Note that E_{t3} obtained in the previous subsection does not satisfy the alignment condition and thus requires message relays. That embedding, however, does have an optimal expansion cost of $2^L/(2^L - 1)$. In contrast, the binary tree embedding presented in the following satisfies the alignment condition, but its expansion cost is not optimal. This demonstrates a trade-off between the expansion cost and the dilation cost for network embeddings in the two-dimensional APPB.

Consider Fig. 9a and assume that we already have an embedding of an s-level binary tree with $N_s = 2^s - 1$ nodes into a two-dimensional APPB of size $a_s \times b_s$. The embedding is assumed to satisfy the alignment condition. That is, all the neighboring nodes in the s-level tree are mapped into the same row or column in the two-dimensional APPB. Using this level s embedding (starting level) as building blocks, the embedding for an $(s + 2)$-level tree is obtained as shown in Fig. 9b. Clearly in this embedding the neighboring nodes are again on the same row or column. A still larger tree is obtained by repeating this modular building procedure until the desired size is achieved. Let us call this embedding E_{t4}. Assuming that in E_{t4} the embedding of an L-level tree, $L = s + 2\sigma$, $\sigma = 0, 1, \ldots$, occupies an area, in number of nodes, equal to A_L in the two-dimensional APPB, we may inductively prove that

$$A_l = 2^{L-s}[A_s + (1 - 2^{-(L-s)/2})b_s].$$

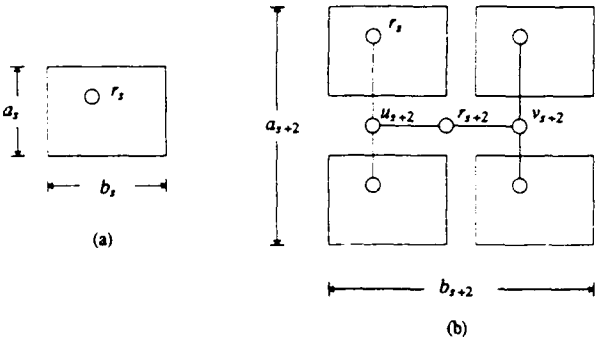


FIG. 9. Modular embedding of binary trees, E_{t4}, in the two-dimensional APPB. (a) A building block in which an s-level binary tree is embedded. (b) Embedding of an $(s + 2)$-level binary tree.

With this result, the expansion cost for the embedding of an L-level tree is

$$C_L = \frac{A_L}{N_L} = \frac{2^{L-s}[A_s + (1 - 2^{-(L-s)/2})b_s]}{2^L - 1} = \frac{2^{L-s}[A_s + (1 - 2^{-(L-s)/2})b_s]}{2^{L-s}(N_s + 1) - 1}.$$

It can be checked that C_L is monotonically increasing with L. However, for large L, the value of C_L asymptotically equals

$$C_{L,\max} = \frac{A_s + b_s}{N_s + 1}.$$

Note that, if $N_s \gg 1$ and $A_s \gg a_s$, the value of C_L simplifies to $C_L \approx A_s/N_s = C_s \geqslant 1$. That is, the expansion cost for the entire embedding is determined by the expansion cost of the building block. Thus low expansion costs may be obtained if the starting building block satisfies $N_s \gg 1$, $A_s \gg b_s$, and $C_s \to 1$. Some examples of building blocks are shown in Fig. 10 with their corresponding expansion cost $C_{L,\max}$. Note that in this modular embedding scheme, as the embedding goes one level higher, the number of levels of the tree increases by 2. Thus if s is even (odd) then L is even (odd). Therefore according to whether the desired level L of the tree is even or odd, the starting level s must be chosen properly.

To determine the control functions for E_{t4}, let r_l be the root at an embedding level l, $l = s + 2, s + 4, \ldots, L$, and (x_l, y_l) be the coordinate, i.e., the row–column position, of r_l in the two-dimensional APPB. Then from Fig. 9b, the coordinate of r_l is

$$(x_l, y_l) = (a_{l-2}, b_{l-2} - 1),$$

where a_l and b_l can be found to be equal to $2^{(l-s)/2}(a_s + 1) - 1$ and $2^{(l-s)/2}b_s$, respectively. Thus,

$$(x_l, y_l) = (2^{(l-s-2)/2}(a_s + 1) - 1, 2^{(l-s-2)/2}b_s - 1).$$

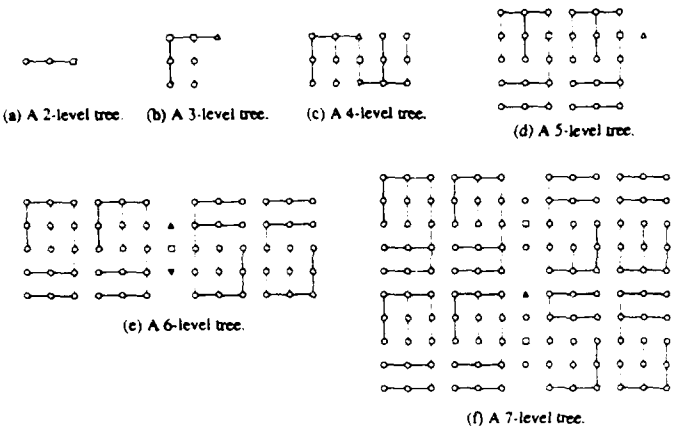


FIG. 10. Example building blocks for the modular embedding of binary trees, E_{t4}, and their corresponding expansion costs $C_{L,\max}$: (a) 1.5, (b) 1.5, (c) 1.25, (d) 1.31, (e) 1.22, (f) 1.12.

Now the control functions can be determined as follows. First, within the building block determine the *wait* functions according to the specific building block in use. Let (x_s, y_s) be the coordinate of r_s, the root in the building block. We then need only determine the *wait* functions for the new nodes which appear as we go to a higher-level embedding. For example, in Fig. 9b when we go from level s to $s + 2$, the new nodes are r_{s+2}, u_{s+2}, and v_{s+2}. By letting $wait_{c,u_l}(r_l)$ be the *wait* function for node r_l to receive a message from child node u_l, we have

$$wait_{c,u_l}(r_l) = y_l - y_{l-2},$$

$$wait_{c,v_l}(r_l) = -(y_l - y_{l-2} + 1),$$

$$wait_{c,r_{l-2}}(u_l) = wait_{c,r_{l-2}}(v_l) = \pm(x_l - x_{l-2}),$$

where the coordinate (x_l, y_l) is as determined previously. These are the *wait* functions for the new parents to receive messages from their children. The *wait* functions for the children to receive messages from these new parents are obtained by recalling that $wait_p = -wait_c$.

Next we show that the binary hypercube of 2^{2k} nodes can also be embedded in a two-dimensional APPB of size $2^k \times 2^k$ such that the alignment condition is satisfied. As in the case of binary trees, the embedding is again modular with the basic module being the binary 2-cube shown in Fig. 11a. A 3-cube embedding is obtained by putting together two such 2-cubes side-by-side as shown in Fig. 11b, and a 4-cube embedding is obtained by putting together two 3-cubes one above the other as shown in Fig. 11c and so on. Note that the nodes in Fig. 11c correspond to the cube nodes of Fig. 5a. In this way the embedding, denoted E_{c2}, of the binary hypercube of the desired size is obtained modularly.

It is observed that in embedding E_{c2}, each row and column is itself a binary k-cube. For example, if we take the column number y as the node id for the nodes in any row x, then row x is a binary k-cube consisting of nodes y, $0 \le y < 2^k$. Let us call each row or column a *subcube*. Then we have 2^{k+1} such subcubes. For each subcube, if we use the column id y (or the row id x) to identify its nodes, all the control functions *step*, *send*, and *wait* are exactly the same as those derived for E_{c1} in the linear APPB. Thus the total communication time for emulating the hypercube can be minimized through overlapped pipelining as presented in the previous section. It can be seen that all the neighboring nodes in the hypercube are mapped to either the same row or the same column in the two-dimensional APPB. Therefore E_{c2} satisfies the alignment condition and thus requires no message relay for communications between neighboring nodes in the hypercube. Finally, since the number of nodes used in the two-dimensional APPB is equal to that of the hypercube, we achieve a minimal expansion cost of unity.

5. BANDWIDTH ANALYSIS

In this section, we evaluate the merit of the pipelined communication structure by comparing it with linear arrays which utilize nearest-neighbor and exclusive access bus interconnections. We evaluate the different models irrespective of the technology used to implement them. In other words, we assume that the transmission rate and the propagation delay are the same for both optical and electronic communication links.

Consider the linear array of n processors with nearest-neighbor connections as shown in Fig. 1b and assume that the physical separation between each pair of neighboring processors is D. Such an array may emulate one cycle of a pipelined bus in a time $n(T_p + T_D)$, where T_D is the propagation time required for a signal to travel the distance D and T_p is the time required to process a message at the sending and the receiving ends of a communication link. T_p includes synchronization, message generation, buffering, and routing. We note that for the cases of interleaved and overlapped pipelining discussed in Section 3.3, at most two messages might be processed in this time. The bandwidth of the nearest-neighbor connected array, B_a, defined as the maximum number of messages that may be transmitted per second, is thus given by

$$B_a = \frac{n}{n(T_p + T_D)} = \frac{1}{T_D}\frac{1}{\rho + 1},$$

where $\rho = T_p / T_D$.

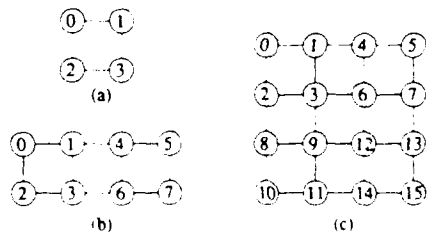


FIG. 11. Modular embedding of binary hypercubes, E_{c2}, in the two-dimensional APPB: (a) 2-cube, (b) 3-cube, (c) 4-cube with the node labeling corresponding to that in Fig. 5a.

For the pipelined linear APPB, the optical distance, D_0, between two consecutive processors should be larger than the message length bwc_g (see Eq. (1) in Section 2). In other words, if $D \geq bwc_g$, we set $D_0 = D$; otherwise D_0 should be made equal to bwc_g (for example, by coiling an optical fiber) so that each processor can inject a message into the bus without collision. Thus, the signal propagation time, T_{D_0}, between two consecutive processors is $\max\{T_D, \alpha T_D\}$, where $\alpha = (bwc_g)/D$. The pipelined bus cycle time is then $T_p + nT_D\max\{1, \alpha\}$. Given that n messages may be transmitted during a pipelined bus cycle, the bandwidth of the pipelined bus is

$$B_p = \frac{n}{T_p + nT_D\max\{1, \alpha\}}. \quad (2)$$

and thus,

$$\frac{B_p}{B_a} = \frac{n(\rho + 1)}{\rho + n\max\{1, \alpha\}}. \quad (3)$$

In Fig. 12a a parametric plot showing the relation between B_p/B_a and ρ is given in terms of n for $\alpha \leq 1$ and $\alpha > 1$. The curve for $\alpha \leq 1$ corresponds to the case where the message length is less than or equal to the physical separation between processors, while the curve for $\alpha > 1$ reflects the case where message length is longer than the physical separation between processors, and thus the optical path has been extended to accommodate the entire message. By taking the limit of Eq. (3) as $\rho \to \infty$, it is clear that, for fixed α and large ρ, the ratio B_p/B_a approaches n. Also, when $\rho = 1$ and $\alpha \leq 1$, we obtain $B_p/B_a \approx 2$. In Fig. 12b we plot B_p/B_a versus ρ for a fixed-size array with $n = 64$ and for several values of α. These plots show that the pipelined bus is more effective for larger values of ρ and smaller values of α.

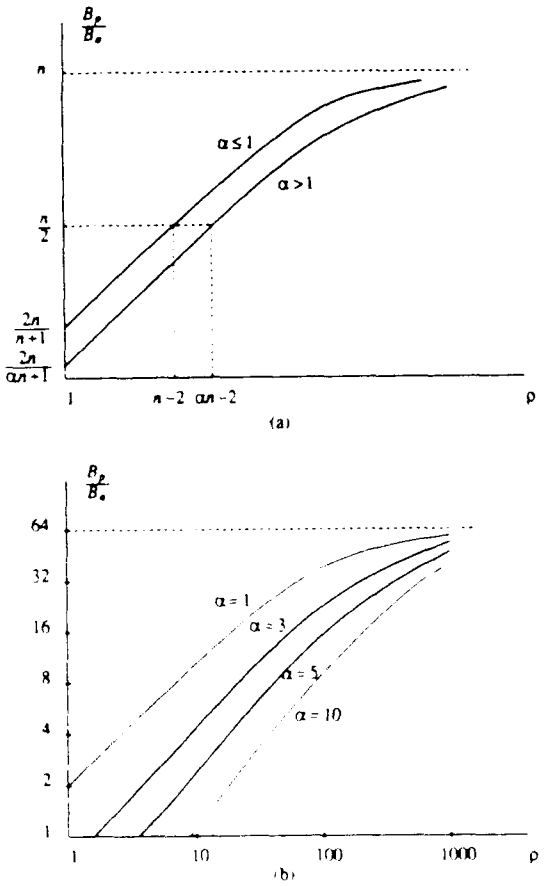


FIG. 12. The ratio, B_p/B_a, of the bandwidth of a pipelined bus to that of a linear array with nearest-neighbor connections as a function of ρ, α, and n. (a) A parametric curve. (b) For a fixed-size system with $n = 64$.

For multiprocessor interconnections, D is determined by placement and routing within VLSI chips, by PC board connections, or by back-plane interconnections. In all cases, D, and therefore T_D, is relatively small. Given that T_p is, at least, on the order of microseconds, the ratio, ρ, of processing to communication times should be much larger than 1 (on the order of 10–1000). Also, with current technology it is reasonable to assume that α is relatively small (between 1 and 10). For example, for board-to-board communications ($D \approx 10$ cm), it is possible to drive an optical communication line at the speed of 10 GHz. Assuming that the speed of light in optical fibers is $c_g = 2 \times 10^8$ m/s, and that each message contains $b = 16$ bits, we obtain $\alpha \approx 3$. The same value of α is obtained if optical communications are implemented on GaAs wafers at 100 GHz and a physical processor separation of 1 cm. Note that the value of α may be reduced if parallel buses are used to reduce b.

Next we compare the bandwidth of a pipelined bus with that of an exclusive access bus. Given that the bandwidth of an exclusive access bus is $B_e = 1/(T_p + nT_D)$, we have

$$\frac{B_p}{B_e} = \frac{n(\rho + n)}{\rho + n\max\{1, \alpha\}}.$$

This shows that as α approaches 1, the pipelined bus can accommodate n messages in the same cycle time as the exclusive access bus. For larger α, the pipelined bus cycle will be stretched to accommodate the length of the messages, and thus, the performance gain due to pipelining will be less than n.

The above analysis is independent of the media used for communication. If optical pipelined buses are to be compared with electronic buses, then the physical constraints on the electronic propagation speed should be taken into account. Specifically, the effect of capacitive loading and mutual inductance on the signal propagation speed (the transmission line effect) should be considered. Thus, message pipelining using electro-optical technology offers a potential for substantially enhancing bandwidth utilization. Further, pipelining techniques will be of increasing effectiveness because this technology offers the capability of generating very short pulses [12, 33], thus reducing w and decreasing α.

6. CONCLUDING REMARKS

We have presented efficient communication architectures which exploit the optical signal's properties of unidirectional propagation and predictable path delays in order to pipeline messages on optical buses. As shown in Section 5, the pipelined model has its merits irrespective of the technology in which it is implemented. Although the presentation in this paper is based on an optical model in which delays inherent in optical fibers serve as slots for space multiplexing, it is possible to use shift registers as buffer memories for these slots [36]. Thus pipelined buses may be implemented in either optics or electronics. However, for the electronic implementation, the signal propagation delay, T_D, will depend on the speed of the shift registers, resulting in a relatively small value for the ratio of processing to communication times, ρ.

We proposed efficient approaches to fundamental message routings including one-to-one, broadcast, semigroup communications, and permutations for the APPB architectures. Such efficient accomplishment of these commonly used message routing patterns can significantly improve the efficiency of many parallel algorithms. We presented here efficient embeddings of the binary trees and hypercube networks. Embeddings for other well-known interconnection networks, including pyramids, shuffle-exchange networks, X-binary-trees [9], and X-quad-trees, have also been obtained [14, 16]. Such efficient embeddings of these well-known communication structures allow all algorithms designed for these structures to be efficiently executed on the APPB architectures. They also allow an APPB to be logically reconfigured as an architecture which is more suitable for a given computation task.

We have not considered in this paper several issues that are relevant to the implementation of the proposed architectures. Such issues include the synchronization of the processors to the accuracy implied by the speed of optics, temporal pulse positioning, optical fanout, and the distribution of optical power in a way that allows the detector at each processor to detect the optical signals correctly. These issues must be addressed with regard to the reliability, scale, and device technology which is appropriate for computing applications. Some of these issues have been presented in [7, 25, 31].

In our experimental work [6, 8, 21] we are investigating the practical limits to these technological concerns. We have shown that three factors, threshold power margin, synchronization error, and coupling ratio, determine the system scale. On the basis of current and near-term technology, our experiments show that synchronization error does not contribute significantly to the bounds of system size. Rather, power distribution effects dominate. Preliminary investigations show that by using off-the-shelf optical components we can currently build linear buses operating at 300 MHz and containing about 100 processors. Using more sophisticated electro-optics (gallium arsenide, custom couplers, and dual level bus structures) we believe that 10-GHz buses of over 400 processors are feasible. Further, we believe that near-term technologies such as fiber amplifiers as well as alternate bus structures will alleviate the power distribution problem.

REFERENCES

1. Bailey, D., and Cuny, J. An efficient embedding of large trees in processor grids. *Proc. 1986 International Conference on Parallel Processing*. IEEE Computer Society, Silver Spring, MD, 1986, pp. 819–822.
2. Batcher, K. E. Design of a massively parallel processor. *IEEE Trans. Comput.* **C-29**, 9 (1980), 836–840.
3. Bokhari, S. H. Finding maximum on an array processor with a global bus. *IEEE Trans. Comput.* **C-32**, 2 (1984), 133–139.
4. Chen, Y. C., Chen, W. T., Chen, G. H., and Sheu, J. P. Designing efficient parallel algorithms on mesh-connected computers with multiple broadcasting. *IEEE Trans. Parallel Distrib. Systems* **1**, 2 (1990), 241–245.
5. Chiarulli, D. M., Melhem, R. G., and Levitan, S. P. Using coincident optical pulses for parallel memory addressing. *IEEE Comput.*, (Dec. 1987), 48–57.
6. Chiarulli, D. M., Levitan, S. P., and Melhem, R. G. Self routing interconnection structures using coincident pulse techniques. *SPIE Proc. International Symposium on Advances in Interconnects and Packaging*, Boston, MA, 1990, Vol. 1390.
7. Chiarulli, D. M., Levitan, S. P., and Melhem, R. G. Optical bus control for distributed multiprocessors. *J. Parallel Distrib. Comput.* **10** (1990), 45–54.
8. Chiarulli, D. M., Levitan, S. P., and Melhem, R. G. Demonstration of an all optical addressing circuit. *Proc. OSA Topical Meeting on Optical Comput.*, Salt Lake City, UT, 1991, pp. 235–238.
9. Despain, A. M., and Patterson, D. A. X-tree: A tree structured multiprocessor computer architecture. *Proc. 5th International Symposium on Computer Architecture*, 1978, pp. 144–151.
10. Duff, M. J. B., Watson, D. M., Fountain, T. J., and Shaw, G. K. A cellular logic array for image processing. *Pattern Recognition* **5** (1973), 229–237.
11. Duff, M. J. B., and Fountain, T. J. *Cellular Logic Image Processing*. Academic Press, New York, 1986.
12. Fujimoto, J., Weiner, A., and Ippen, E. Generation and measurement of optical pulses as short as 16 fs. *Appl. Phys. Lett.* **44** (1984), 832–834.
13. Gordon, D., Koren, I., and Silberman, G. Embedding tree structures in VLSI hexagonal arrays. *IEEE Trans. Comput.* **C-33**, 1 (1984), 104–107.
14. Guo, Z. Array processors with pipelined busses and their implication in optically and electronically interconnected multiprocessor architectures. Ph.D. thesis, Department of Electrical Engineering, University of Pittsburgh, 1991.
15. Guo, Z., Melhem, R. G., Hall, R. W., Chiarulli, D. M., and Levitan, S. P. Array processors with pipelined optical busses. *Proc. 3rd Symposium on Frontiers of Massively Parallel Computation*, 1990, pp. 333–342.
16. Guo, Z., and Melhem, R. G. Embedding pyramids in array processors with pipelined busses. *Proc. International Conference on Application Specific Array Processors*, 1990, pp. 665–676.
17. Hunt, D. J. The ICL DAP and its application to image processing. In Duff, M. J. B., and Levialdi, S. (Eds.) *Languages and Architectures for Image Processing*. Academic Press, San Diego, CA, 1981.
18. Jrad, A. M., and Hall, R. W. The OFC enhanced mesh architecture: A performance study. *Proc. 1987 Workshop on Computer Architecture for Pattern Analysis and Machine Intelligence*, 1987, pp. 184–191.

19. Jrad, A. M., and Hall, R. W. Orthogonal fast channels: An enhanced mesh architecture. *Proc. 1987 International Conference on Parallel Processing*. IEEE Computer Society, Silver Spring, MD, 1987, pp. 828–831.

20. Kawasaki, B. S., Hill, K. O., and Lamont, R. G. Biconical-taper single-mode fiber coupler. *Opt. Lett.* **6**, 7 (1981), 327–328.

21. Levitan, S. P., Chiarulli, D. M., and Melhem, R. G. Coincident pulse techniques for multiprocessor interconnection structures. *Appl. Opt.* **29**, 14 (1990), 2024–2033.

22. Melhem, R. G., Chiarulli, D. M., and Levitan, S. P. Space multiplexing of waveguides in optically interconnected multiprocessor systems. *Comput. J.* **32**, 4 (1989), 362–369.

23. Miller, R., and Stout, Q. F. Mesh computer algorithms for computational geometry. *IEEE Trans. Comput.* **C-38**, 3 (1989), 321–340.

24. Misra, M., and Prasanna-Kumar, V. K. Efficient VLSI implementation of iterative solutions to sparse linear systems. Tech. Rep. IRIS 246, University of Southern California, 1988.

25. Nassehi, M., Tobagi, F., and Marhic, M. Fiber optic configurations for local area networks. *IEEE J. Selected Areas Commun.* **SAC-3**, 6 (1985), 941–949.

26. Nassimi, D., and Sahni, S. Data broadcasting in SIMD computers. *IEEE Trans. Comput.* **C-30**, 5 (1981), 101–107.

27. Nath, D., Maheshwari, S. N., and Bhatt, P. C. P. Efficient VLSI networks for parallel processing on orthogonal trees. *IEEE Trans. Comput.* **C-32**, 6 (1983), 569–581.

28. Prasanna-Kumar, V. K., and Eshaghian, M. M. Parallel geometric algorithms for digitized pictures on mesh of trees. *Proc. 1986 International Conference on Parallel Processing*. IEEE Computer Society, Silver Spring, MD, 1986, pp. 270–273.

29. Prasanna-Kumar, V. K., and Raghavendra, C. S. Array processor with multiple broadcasting. *J. Parallel Distrib. Comput.* **4** (1987), 173–190.

30. Prasanna-Kumar, V. K., and Reisis, D. Image computations on meshes with multiple broadcast. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-11**, 11 (1989), 1194–1202.

31. Prucnal, P., Blumenthal, D., and Perrier, P. Self routing photonic switching demonstration with optical control. *Opt. Engrg.* **26**, 5 (1987), 473–477.

32. Raghavendra, C. S., and Prasanna-Kumar, V. K. Permutations on Illiac-IV type networks. *IEEE Trans. Comput.* **C-37**, 7 (1986), 662–669.

33. Shank, C. The role of ultrafast optical pulses in high speed electronics. In Morou, G., Bloom, D., and Lee, C. (Eds.). *Picosecond Electronics and Opto-Electronics*. Springer-Verlag, New York, 1985.

34. Singh, A. Near optimal embedding of binary tree architecture in VLSI. *Proc. 8th Symposium on Distributed Computing Systems*, 1988, pp. 86–93.

35. Stout, Q. F. Mesh connected computers with broadcasting. *IEEE Trans. Comput.* **C-32**, 9 (1983), 826–830.

36. Tanenbaum, A. S. *Computer Networks*. Prentice-Hall, Englewood Cliffs, NJ, 1981.

37. Thompson, C. D., and Kung, H. T. Sorting on a mesh-connected parallel computer. *Commun. ACM* **20**, 4 (1977), 263–271.

38. Tobagi, F., Borgonovo, F., and Fratta, L. Expressnet: A high-performance integrated-services local area network. *IEEE J. Selected Areas Commun.* **SAC-1**, 5 (1983), 898–912.

39. Ullman, J. D. *Computational Aspects of VLSI*. Computer Science Press, Rockville, MD, 1984.

40. Whalen, M. S., and Wood, T. H. Effectively nonreciprocal evanescent-wave optical-fibre directional coupler. *Electron. Lett.* **21**, 5 (1985), 175–176.

Received April 6, 1990; accepted January 11, 1991

ZICHENG GUO is finishing his Ph.D. in the Department of Electrical Engineering at the University of Pittsburgh. His current research interests include parallel computer architectures and algorithms, optical communications in multiprocessor networks, and image computation and pattern recognition.

RAMI G. MELHEM is an associate professor of computer science at the University of Pittsburgh. He received a B.E. in electrical engineering from Cairo University, Egypt, in 1976, an M.S. in mathematics/computer science from the University of Pittsburgh in 1981, and a Ph.D. in computer science from the University of Pittsburgh in December 1983. He has been an assistant professor of computer science at Purdue University from 1984 to 1986 and at the University of Pittsburgh from 1986 to 1989. His research interests include optical computing, parallel systems, fault-tolerant systems, and the application of large computational arrays to scientific problems.

RICHARD W. HALL received the B.S.E. degree in electrical engineering from The Evening College of the Johns Hopkins University in 1969 as part of the Westinghouse–Johns Hopkins Awards Program and the M.S. and Ph.D. degrees in electrical engineering from Northwestern University in 1971 and 1975, respectively. He joined the Department of Electrical Engineering at the University of Pittsburgh in 1975 and is currently an associate professor in that department. His current research interests are in the study of parallel algorithms and architectures for visual information processing.

DONALD M. CHIARULLI is an assistant professor of computer science at the University of Pittsburgh. He received a B.S. degree in physics from Louisiana State University in 1976, an M.S. degree in computer science from Virginia Polytechnic Institute in 1979, and a Ph.D. in computer science from Louisiana State University in 1986. From 1979 to 1983, he was President of Dataflex Services Inc., a consulting and software development firm. While at Louisiana State he was responsible for the design and construction of The Factoring Machine, a reconfigurable VLIW machine for factoring large numbers. Dr. Chiarulli's current research interests include hybrid optical/electronic computer architecture, optical interconnects, VLSI design, and parallel computation. He is a member of the IEEE Computer Society, ACM, SPIE, and the Optical Society of America.

STEVEN P. LEVITAN is the Wellington C. Carl Assistant Professor of Electrical Engineering at the University of Pittsburgh. He received the B.S. degree from Case Western Reserve University (1972) and his M.S. (1979) and Ph.D. (1984) degrees, both in computer science, from the University of Massachusetts, Amherst. He worked for Xylogic Systems, designing hardware for computerized text processing systems, and for Digital Equipment Corp. on the Silicon Synthesis project. He was an assistant professor from 1984 to 1986 in the Electrical and Computer Engineering Department at the University of Massachusetts. In 1987 he joined the electrical engineering faculty at the University of Pittsburgh. Dr. Levitan's research interests include computer-aided design for VLSI, parallel computer architecture, parallel algorithm design, and VLSI design. He is a member of the IEEE Computer Society, ACM, SPIE, and OSA.

Time-Division Optical Communications in Multiprocessor Arrays

Chunming Qiao and Rami G. Melhem

Department of Computer Science
University of Pittsburgh, Pittsburgh, PA 15260

Abstract

An optical communication structure for multiprocessor arrays that exploits the high communication bandwidth of optical waveguides is proposed. The structure takes advantage of two properties of optical signal transmissions on waveguides. Namely, unidirectional propagation and predictable propagation delays per unit length. Two novel time-division multiplexing approaches are proposed for non SIMD environments to obtain a communication bandwidth comparable to that of message pipelining in SIMD environments. Analysis and simulation results are given to evaluate the communication effectiveness of the system. A clock distribution method is also proposed to address potential synchronization problems. Finally, feasibility issues with current and future technologies are discussed.

1. Introduction and background

Optical interconnections for communication networks and multiprocessor systems including both free-space and guided wave [7, 8, 9, 21, 22] approaches have been studied extensively in the literatures. In this paper, we propose a waveguide interconnection system with time-division communications.

Time-division communications are especially useful in optical interconnected systems where high communication bandwidth can be exploited. Optical pulse transmissions on a waveguide have two distinct properties from electronic signal transmissions, namely unidirectional propagation and predictable propagation delays per unit length. In a multiprocessor system connected with an optical waveguide (or bus), relationships between the spatial and temporal positions of transmitted pulses can be established. For example, if two processors transmit a pulse on the waveguide at the same time, the difference between arrival times of these two pulses at any checkpoint downstream, is equal to the propagation delays between the two processors. In other words, the spatial separation of the two processors determines the temporal separation between the pulses they transmit. By arranging spatial separations and controlling transmission (or receiving) times of processors, time-division multiplexings (or demultiplexings) can be done without using multiplexers (or demultiplexers).

Several time-division switching approaches can be applied in a multiprocessor system connected by optical buses. In the first approach, each processor is assigned a fixed time slot and transmits or receives a message during that particular time slot. A sequence of time slots formed on the transmitting segment of a bus is rearranged via a time-slot interchanger [21, 24] and then forwarded to the receiving segment. Each time slot of the output sequence contains a message destined to the processor corresponding to that slot. In the second approach, each processor is assigned a fixed *transmitting* time slot. A sequence of time slots formed on the transmitting segment is directly forwarded to the receiving segment without interchanging time slots. Instead of assigning a fixed receiving time slot to each processor, a *SIMD* environment is assumed where each processor knows which processor is sending a message to it and knows the time slot that contains the message. Since there is a one-to-one mapping between a source processor and a time slot, we call this approach time-division source-oriented multiplexing (or TDSM). It has also been referred to as bus pipelining in [7, 16].

TDSM may also be applied in a non *SIMD* environment, where source processors are not known to the destination processors. In this case, each message should contain address information so that each processor will be able to receive messages upon address decodings. Another approach, which is also applicable in a non *SIMD* environment, assigns a fixed *receiving* time slot to each processor. Each message is transmitted during the receiving time slot assigned to its destination processor. Since there is a one-to-one mapping between a destination processor and a time slot, we call this approach time-division destination-oriented multiplexing (or *TDDM*). Since each destination only has one dedicated receiving time-slot, contentions can occur if

1991 ACM 0-89791-459-7/91/0644 $01.50

several sources want to send messages to the same destination. One way to ensure exclusive access of a timeslot is to use a reservation scheme.

In this paper, we will discuss TDSM and TDDM approaches and apply the combination of these two in our system design. We use coincident pulse techniques [3, 13, 20] to encode address information that is contained in messages. In order to explain coincident pulse addressing, we consider an optical bus connected linear array of N processors, as shown in *Figure 1*.

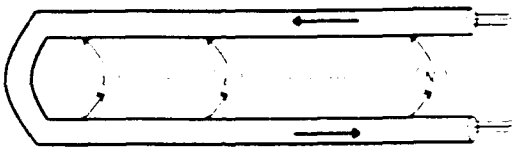
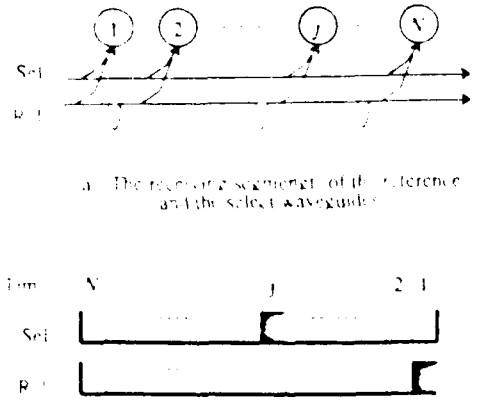


Figure 1. A linear optical array

Each processor transmits on the upper half segment of a bus and receives from the lower half segment. The optical bus consists of three waveguides, one for carrying messages (the *message waveguide*) and two for carrying address information (the *reference waveguide* and the *select waveguide*). Messages are organized as *message frames*, which have a certain fixed length. The propagation delay on the reference waveguide is the same as that on the message waveguide but not the same as that on the select waveguide. A fixed amount of additional delay, which we show as loops in *Figure 2*, is inserted onto the reference waveguide and the message waveguide.



a. The receiving segments of the reference and the select waveguides

b. an address frame relative to a reference pulse

Figure 2. Unary coincident pulse addressing

Let w be the pulse duration in seconds, and let c_b be the velocity of light in the waveguides. Define a unit time to be the spatial length of a single optical pulse, that is $w \times c_b$. Starting with the fact that all three waveguides have equal intrinsic propagation delays, we add one time unit delay between any two processors on the receiving (lower half) segments of the reference waveguide and the message waveguide as shown in *Figure 2 (a)*. Since there are no changes on the transmitting (upper half) segments of any waveguide and since the message waveguide has exactly the same length as the reference waveguide, *Figure 2(a)* shows only the receiving segments of the select waveguide and the reference waveguide.

A source processor sends a reference pulse and a select pulse at appropriate times, so that after these two pulses propagate through their corresponding waveguides, a coincidence of the two occurs at the desired destination. The source processor also sends a message frame which propagates synchronously with the reference pulse. Whenever a processor detects a coincidence of a reference pulse and a select pulse, it reads the message frame. More specifically, Let t_{ref} be the time when processor i transmits its reference pulse and $t_{sel}(j)$ be the time when it transmits a select pulse. These two pulses will coincide at processor j if and only if

$$t_{sel}(j) = t_{ref} + j \qquad (1.1)$$

where $0 \le i, j < N$. This means that for a given reference pulse transmitted at time t_{ref}, the presence of a select pulse at time $t_{ref} + j$ will address processor j while the absence of a select pulse at that time will not. In essence, the address of a destination processor is unary encoded by the source processor using the relative transmission time of a reference pulse and a select pulse.

Call the duration of each pulse, w, a pulse slot. A sequence of pulse slots on the select waveguide, each with either the presence or the absence of a select pulse relative to a given reference pulse, is called an *address frame*. *Figure 2 (b)* shows a snapshot of a reference pulse and an address frame just after they have been transmitted. At the transmission time, the select pulse is j units behind the reference pulse. Since the reference pulse will be delayed by one unit each time it propagates through a processor on the receiving segment of the reference waveguide, these two pulses will coincide at processor j.

With the above unary addressing, an address frame contains N pulse slots and is essentially a time multiplexed sequence of pulse slots, each corresponding to a destination processor. As the address frame propagates through the receiving segment of the bus,

demultiplexings of each pulse slot is performed with respect to a reference pulse via unit delays added on the reference waveguide. Address decoding, which could be a bottleneck with traditional addressing mechanisms, is done through the detection of a coincidence at the destination.

Define a *packet* to be a collection of information including a message frame, an address frame and a reference pulse. Let P be the length of a packet in time units and D be the spatial separation of any two adjacent processors on an optical bus. Because an address frame length is N time units, we have the condition $P \geq N$. With TDSM communications, if all processors transmit packets simultaneously, then the condition $D \geq P$ has to be satisfied in order to prevent packet overlappings. This condition, together with the condition that $P \geq N$, limits the system size N. In addition, the address frame length could be longer than the message frame length if N is large, resulting in inefficiency. Another factor that limits the system size relates to the power distribution. Specifically, the system size is limited by the minimum powers that can be detected at the last processor in a linear system [4].

Because of these shortcomings in a linear system, we propose a two dimensional $n \times n$ array called *ASOS* for *Array structure with Synchronous Optical Switches*. In Section 2, we present the system configuration of the proposed *ASOS*. We also show how communications in the *ASOS* are done with TDDM, TDSM and the combination of the two. In Section 3, we give analysis and simulation results that relate to communication effectiveness of the system. In Section 4, we address the potential synchronization problem and propose a global clock distribution model as it relates to the packet size limitation. And finally in Section 5, we determine the merits of the design and conclude the paper.

2. Array structure with synchronous optical switches

2.1. System configuration

The system configuration of an Array structure with Synchronous Optical Switches (or *ASOS*) is shown in *Figure 3*. Processors in the *ASOS* are connected with a set of folded horizontal (row) buses and vertical (column) buses, each consisting of three waveguides as in a linear system. All row buses are assumed to be identical, so are all column buses. During the course of the following presentation, rows are numbered from top to bottom and columns and processors are numbered from left to right.

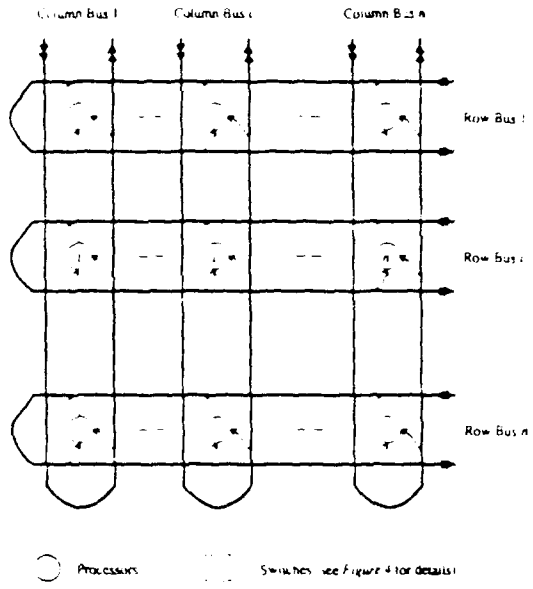


Figure 3. System configuration of the *ASOS*

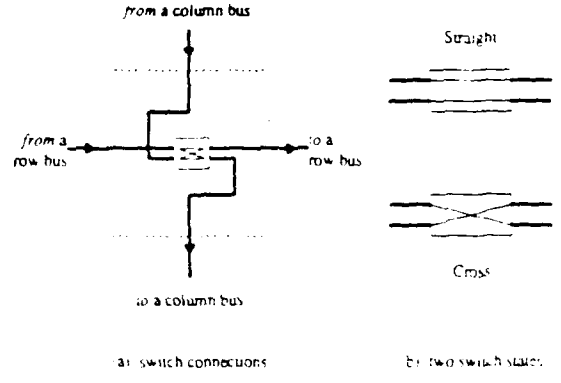


Figure 4. Connections of a switch

A processor can transmit on the upper segment of a row bus while receiving from the lower segment of a row bus and the right segment of a column bus concurrently. As shown in *Figure 3*, an optical switch is placed at each intersection of the lower segment of a row bus and the left segment of a column bus. The switch connects a row and a column bus as shown in *Figure 4(a)*. Each switch is a 2×2 optical device which can be in one of the two states: straight or cross, as shown in *Figure 4 (b)*. If a switch is in the straight state, a message packet propagating on a row bus will continue propagating on it. However, if a switch is in the cross state, a message packet propagating on a row bus will be switched over to a column bus. Several kinds of optical switches can be used for this purpose [1, 2], and switch control is straightforward in *ASOS*, as will be discussed

in the next section. A global clock is used for synchronization purposes and it is assumed that all processors and switches receive identical copies of the synchronization clock. How synchronizations can be retained without the above assumption is discussed in Section 4. Note that, synchronizing the communication structure does not mean that the processors have to execute in a synchronized mode. Each processor may execute at its own pace and submit messages to the communication structure independently. The delivery of messages, however, is controlled by a synchronized structure.

2.2. Row and column communications

Communication between processors at the same row, which we call *row communication*, use each row bus for transmitting and receiving messages with switches set to the straight state. Since packets can not be transmitted directly on column buses, communication between processors at different row, which we call *column communication*, use both row and column buses with switches connecting them set to the cross state. Row communication and column communication alternate, in what we call *row phases* and *column phases* respectively. Switches alternate their two states accordingly. All switches are set to the same state simultaneously.

Let the optical path length between any two adjacent processors (or two adjacent switches) on a row bus be D units long. In order to allow simultaneous transmissions or switchings of packets without overlappings, we require $D \geq P$. We further let the folded optical path length on a row bus be D units (see *Figure 5*). Let $T = (2n - 1)D$ be the end-to-end propagation delays of a row bus. Similarly, let the optical path length between two adjacent processors (or two adjacent switches) on a column bus be D units long and the end-to-end propagation delays of a column bus be T units.

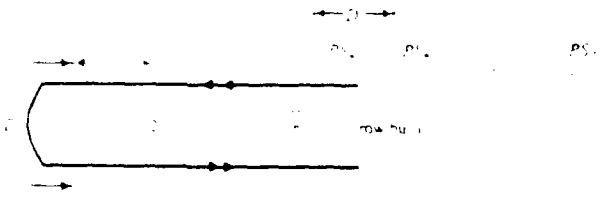


Figure 5. Originating an imagined train

Time-division multiplexing in the *ASOS* can be best explained using the *train loading* model described below. Each time-slot is P units long and is called a *packet slot* (PS). Imagine that a train of n packet slots is originated on a row bus, as shown in *Figure 5*. When the beginning of the train is at processor n, a communication phase begins. Packet slots in a train are numbered as $n, n-1, \ldots, 1$ from left to right. Two adjacent packet slots in a train are separated by D units. That is, there is a gap of $D - P$ units between two slots. We call a train originated at the beginning of a row phase or a column phase a *row train* or a *column train* respectively. Note that column buses, switches and taps are omitted from the figure. Further, a row or a column train should be regarded as three separate trains on the message, select and reference waveguides respectively. Nevertheless, when there is no confusion, we will view these three trains as one on either a row or a column bus.

Assume that a train is originated at the beginning of a communication phase (at time t_r). Let $P\_Arr(i, p)$ be the time that the PS_i arrives at processor p on the upper segment of the bus. We have

$$P\_Arr(i, p) = t_r + (n - i)D + (n - p)D$$
$$= t_r + T - (i + p - 1)D \qquad (2.1)$$

The TDSM is used in a row phase. Each PS_i in a row train is assigned as a transmitting slot to processor i. Each processor can send out one packet during each row phase and each packet contains unary coincident pulse addressing information. More specifically, let t_r be the time when a row phase begins, processor i transmits a packet, if it has one, at time $P\_Arr(i, i)$. That is, as the train propagates through the upper segment of a row bus, processor i loads PS_i with its packet.

One advantage of using TDSM is that a processor can receive more than one message frames in a single row phase, a capability which is usually referred to as m-to-1 communications. Another advantage with TDSM is that a processor can send out a message frame to several destinations in a single row phase efficiently, a capability which is usually referred to as multicasting. This is done by multiplexing several select pulses, each corresponding to one destination, in an address frame [13, 20].

Since a train is nD units long, the last packet slot of the train, namely PS_1, leaves processor n on the upper segment at the time $t_r + nD$. At that time, the row phase ends and a column phase begins with a column train originated. In a column phase, TDDM is used. Each PS_i of a column train is assigned as a receiving slot to the i-th switch at a row bus. That is, a packet transmitted during PS_i is to be switched to column bus i. If more than one processor want to send packets to the same column bus, packet slot contentions will occur. Such contentions are solved by using packet slot reservation schemes as will be discussed later. For now, assume that reservations of every packet slots have been

done and therefore only one processor will transmit a packet during any specific packet slot. Note however, that a processor could send several packets, each to a different column bus, if it has reserved the corresponding packet slots.

During a column phase, each processor loads packet slots that it has reserved while the train propagates through the upper segment of a row bus. More specifically, if processor p has reserved PS_i, it will transmit a packet which is to be switched to the column bus i at time $P\_Arr(i, p)$. T units after a column phase begins, each packet slot of a column train will arrive at its corresponding switch simultaneously as shown in *Figure 6*. Every switch is set to the cross state for P units to switch a packet over to a column bus, with which the destination processor of the packet is connected. That is, if a column phase begins at time $t_c = t_r + nD$, then all switches will be in the cross state during the time period from $t_c + T$ to $t_c + T + P$. Note that, we have assumed a negligible switching time. In reality, the switching time ranges from a few hundreds of picoseconds to a few nanoseconds. If S is the time needed for a switch to change from one state to the other, it is sufficient to let $D = P + S$ and let switches start switching to the cross state at the time $t_c + T - S$. Each switch will stay in the cross state for only P units and start switching to the straight state at the time $t_c + T + P$. Note that, in a non SIMD environment, TDDM has the advantage of timely and orderly delivering messages to passive destinations, such as switches, without address information.

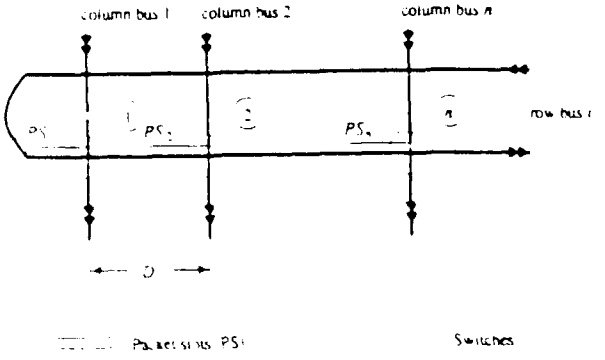


Figure 6. Simultaneous switchings of packet slots

Before a column train is switched, a new row phase begins. A column phase ends as soon as the last packet slot of a column train, namely PS_1, leaves processor n on the upper segment. That is, a column phase also takes nD unit time. A row train will be originated and propagating on the upper segment while a column train is propagating on the lower segment. Note that the beginning of PS_n of the row train is also $D - P$ units away from the end of PS_1 of the preceding column train. Therefore, even with non-negligible switching time S, all switches can be set back to the straight state while the row train propagates on the lower segment of a bus.

When switches on a row bus are set to the cross state, switches at other row buses are also set to the cross state simultaneously. Since two adjacent switches on a row bus are D units apart, packet switched from different row buses will not overlap with each others on a column bus. Rather, these packets will form a train on the left segment of a column bus, as shown in Figure 7.

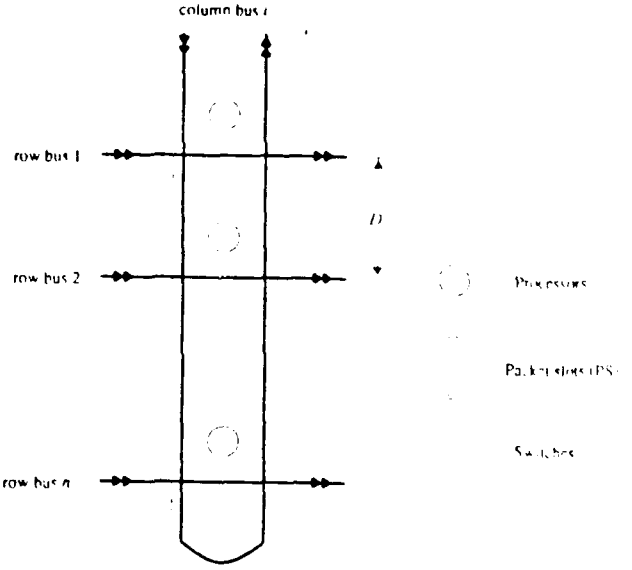


Figure 7. A packet train formed on a column bus

After a packet is switched, it will leave the left segment and propagate on the right segment of a column bus by the time switches are set to the cross state again. That is, no packets on any column bus will be switched back to a row bus. Every packet contains an address frame encoded using coincident pulse techniques and a coincidence will occur at the destination processor, as will be described in the next subsection. In essence, TDDM is used to switch a packet onto a proper column bus and the destination processor on that column bus is then addressed using the coincident pulse techniques.

2.3. Coincident pulse addressing

As mentioned in the previous section, coincident pulse addressing are used in both row and column communications. Each row bus is similar to a bus used in a linear system (refer to *Figure 1* and *Figure 2(a)*). Each

column bus can be viewed as a row bus rotated 90° degree anticlockwise. That is, on the right (receiving) segment of a column bus, one unit delay is added between any two adjacent processors on the message waveguide and the reference waveguide.

A packet in a row train propagates only on a row bus, that is, its reference pulse will encount added delays only on the row bus. To cause a coincidence of the reference pulse and a select pulse at processor j of that row bus, the relative transmission times of the two pulses should satisfies equation (1.1). A packet in a column train, however, propagates on a row bus and a column bus. A reference pulse of such a packet will encounter additional unit delays on the right half segment of that column bus before arriving at a destination processor. More specifically, if a packet in a column train is destined to a processor at row i and column j, its reference pulse will first encount j added unit delays on a row bus and then $n - i$ added unit delays on a column bus. Therefore, to cause a coincidence of the reference pulse and a select pulse at that destination, the relative transmission times of these two pulses should satisfy the following equation :

$$t_{sel} = t_{ref} + j + (n - i) \qquad (2.2)$$

Since we have $1 \le j + (n - i) \le 2n - 1$, the length of an address frame should be $2n - 1$ units long and therefore $D \ge P \ge 2n - 1$. Note that, based on the row and column number of destination processors, each processor can select appropriate packet slots to send packets with the address information, and routings are accomplished through propagations and possible switchings of packet slot trains.

2.4. Packet slot reservation scheme

As mentioned earlier, when loading a column train using TDDM, packet slot reservations are required to resolve contentions. Reservations can be made concurrently with message transmissions using separate folded waveguides, which we call *reservation waveguide*, one at each row, as in Figure 8.

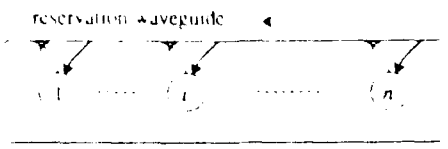


Figure 8. A reservation waveguide

There are n packet slots to be reserved for each column train before the train is loaded in a column phase. To reserve packet slots for a column train, a corresponding train, which we call a *reservation train*, is originated on a reservation waveguide. The reservation train also consists of n reservation slots with the same separations between two slots as in a column train. Each of the reservation slot in a reservation train is used to arbitrate the reservation of the corresponding packet slot in a column train. The time period from the origination of a reservation train to the end of reservation operations on its n reservation slots is called a *reservation cycle*. A reservations cycle can overlap with a row phase and a column phase as long as the reservation results will be available in a corresponding column phase.

Three reservation schemes have been studied. The simplest reservation scheme is the linear priority scheme in which processor upstream are given higher priorities than processors downstream. When competing for a reservation of a packet slot, the processor that has the highest priority among all competing processors will succeed while others will fail. For example, in Figure 8, processor n is given the highest priority and processor 1 the lowest priority. After a reservation cycle begins, each processor monitors reservation slots of a reservation train propagating on the upper segment of the reservation waveguide. If processor p wants to reserve packet slot i, it transmits pulses to the left while also detecting pulses coming from the right during reservation slot i. If no pulses are detected in that period, processor p has succeeded in making a reservation for PS_i since no processors upstream have attempted to reserve it. Otherwise, processor p fails and a processor upstream has succeeded in reserving PS_i. Reservation operations will finish as soon as the train leaves the upper segment. Assuming that a reservation slot equals to a packet slot, a reservation cycle will take T units.

The problem of the simple linear priority scheme is the possibility of starvations. Processors downstream with lower priorities may be indefinitely blocked because some higher priority processors at the right keep making reservations. In the restrained linear priority scheme, linear priorities among processors are still enforced but we require a processor that succeeded in reserving a packet slot in a previous cycle to refrain from making another reservation for the same packet slot until after an *idle* cycle in which no processors reserve that slot. Therefore, any processor would be able to succeed in making a reservation for a packet slot within n cycles and no starvations are therefore possible.

Nevertheless, using this scheme, a higher priority processor still has more chances of success than lower priority processors when competing for reservations. The third scheme fully employs round-robin (or cyclic

polling [11]) A processor currently with the highest priority for a packet slot among all competing processors will succeed in making a reservation. In the next reservation cycle, it will become the lowest priority one when competing for the same packet slot. Meanwhile, the processor with the priority next to the one succeeded in this last cycle will have the highest priority and all other processors will adjust their priorities accordingly in a cyclic order.

Implementation issue of the three reservation schemes are discussed in details in [19], which also includes simulation results regarding the fairness of the three reservation schemes.

3. System bandwidth and packet delays

Let B_{max} be the maximum transmission rate at which a processor can drive an optical bus. Then the maximum bandwidth of a row bus is B_{max} and the maximum bandwidth of the $n \times n$ ASOS is nB_{max}. Since during each packet slot which is $D = P + S$ units, at most P unit messages are transmitted. Therefore, the maximum efficiency, Θ, is

$$\Theta = \frac{P}{P+S} \tag{3.1a}$$

And the maximum bandwidth achievable, B_a, is

$$B_a = n \times B_{max} \times \Theta \tag{3.1b}$$

Assume that on average, each processor generates L_r packets during each row phase and L_c packets during each column phase, where $0 \le L_r, L_c \le 1$. Denote the average number of packets generated per packet slot by L_s. Then $L_s = \frac{L_r + L_c}{2}$. The average throughput, or effective bandwidth in a row phase, B_r, is

$$B_r = L_r \times B_a \tag{3.2}$$

If we assume that the destinations of generated packets are uniformly distributed among n column buses, the *maximum* effective bandwidth in a column phase, B_c, may be approximated by

$$B_c = L_c \times B_a \tag{3.3}$$

Note that this maximum bandwidth in a column phase may not be achievable due to packet slot contentions. Therefore, given fixed average communication load L_s, mappings that distribute more loads to row communications and less loads to column communications will improve the effective system bandwidth.

By taking the average of equation (3.2) and equation (3.3), the effective system bandwidth of an ASOS, denoted by B_e, is

$$B_e = \frac{B_r + B_c}{2} = \frac{n(L_r + L_c)B_{max}P}{2(P+S)} \tag{3.4}$$

Note that, from equations (3.1), the switching speed S, relative to the packet length P, largely determines the system efficiency and bandwidth.

Packet slot contentions in column communications not only can decrease the effective bandwidth but also can introduce delays for packets. In a column phase, a processor will not be able to send out a packet unless it has succeeded in reserving the corresponding packet slot. The packet that can not be sent out due to an unsuccessful reservation has to be delayed until a future column phase. Define packet delays to be the number of column phases that a packet has been delayed due to unsuccessful reservations of the corresponding packet slot. Further, assume that during each column phase, the number of packets that each processor generates is a poison process with mean rate λ where $\lambda < 1$. The destinations of these packets are evenly distributed among n columns. That is, on the average, out of λ packets generated by a processor in each column phase, only $\lambda_i = \frac{\lambda}{n}$ packets are destined for column bus i.

We first examine the average packet delays using the round-robin reservation scheme. A natural analytic model to use is the model with multiple queues and a single server. Each processor is viewed as a station with an ideally infinite queue. The server selects a station to serve in the round-robin fashion. The service time is a constant unit time (which is one column phase). In [23], a similar model is analyzed where the *reply interval*, defined as the time for a server to switch from one station to another, is assumed to be non-zero. For models with zero reply interval such as ours, only approximate analysis are given [12]. However, if we view the above model with multiple queues as a single queue, $M/D/1$ model with the *FCFS* (First-Come First-Serve) policy, we will have the same result for average packet delays for both models. In fact, with the same assumptions about message arrival rate and service rate in both models, the statistical characteristics of the unfinished work in both models should be identical. In other words, the total number of packets remaining, hence the queue length in both models, is independent of the service policy used. It follows from the Little's law [14] that the average packet delays, denoted by E_r, is also independent of the policy used. That is, the average packet delays using round-robin scheme should be the same as that of a $M/D/1$ model. Namely [5, 10]

$$E_r = \frac{n\lambda_i}{2(1 - n\lambda_i)} = \frac{\lambda}{2(1-\lambda)} \tag{3.5}$$

It is also straightforward to have the same analysis result for average packet delays using linear priority scheme. However, we are unable to analyze average packet delays using the restrained linear priority scheme. *Figure 9* shows simulation results of average packet delays vs. arrival rates λ for all three different reservation schemes. We note that the results shown for both round-robin and linear priority schemes conform to the theoretical values as outlined in equation (3.5). However, the restrained linear priority scheme has longer average packet delays since some idle cycles are artificially introduced in this scheme.

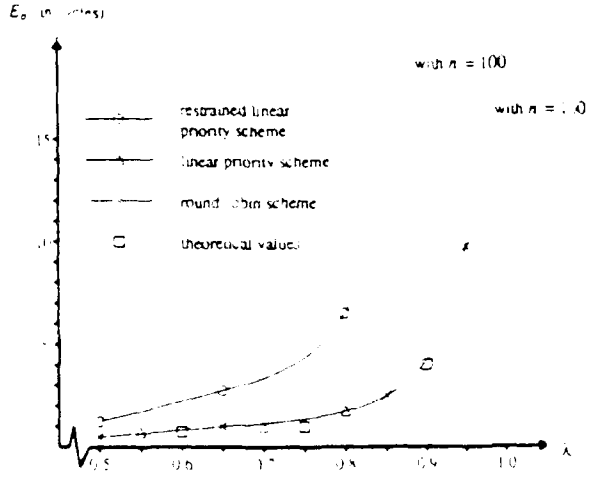


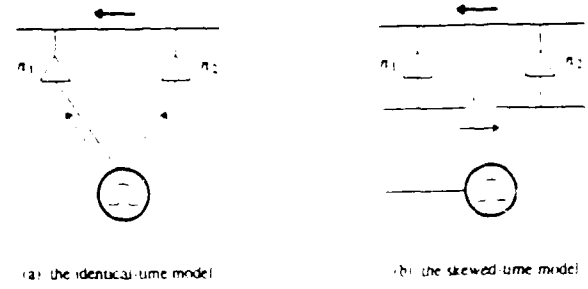
Figure 9. Average packet delays

If either the linear priority or the round-robin scheme is used, average packet delays in column communications are only about two column phases when the load reaches 80% of the capacity (this is equivalent to the case when $\lambda = 0.8$). Note however, that according to the definition of the packet delays, packets in row communication have zero packet delays.

4. Clock distribution and packet size limitation

One of the major concerns in designing a large system is the synchronization problem. So far, we have assumed that all processors and switches are connected to a global clock with separate waveguides (called clock waveguides) of equal length. Therefore, for communication purposes, all processors and switches virtually share an identical global time. More specifically, let T_g be a global time, and let $PT_L(r, c)$ and $ST_L(r, c)$ be a local time of the processor and the switch respectively, at row r and column c. We have assumed that at any instance, $PT_L(r, c) = ST_L(r, c) = T_g + C$ for all r and c, where C is a constant. We call such a clock distribution model *identical-time* model.

For example, consider a system with two nodes, denoted by n_1 and n_2. Let $T_L(1)$ and $T_L(2)$ be their local time respectively. To perform an operation *simultaneously* at a given instance T_g, each node starts the operation when its local time equals T_g. Here, an operation could be the transmission of a packet if the nodes are processors, or could be the change from one state to the other if the nodes are switches. In the identical-time model shown in *Figure 10(a)*, two separate clock waveguides with equal length are used to connect two nodes with the global clock. Since $T_L(1)$ is always identical to $T_L(2)$, the two nodes will start at the same time to perform a simultaneously operation on two packets. If these two nodes are separated by D units on a waveguide and a packet has a length of P units, the condition $P \leq D$ is necessary to prevent the two nodes from performing the same operation on the same packet. For instance, if these two nodes are processors, the above condition prevents overlappings of transmitted packets. If these two nodes are switches, the above condition prevents a packet from being partially switched by two switches.



(a) the identical-time model (b) the skewed-time model

Figure 10. Two clock distribution models

Another model, which we call *skewed-time* model, is shown in *Figure 10(b)*. Only one clock waveguide is used to connect these two nodes to the global clock. It is assumed that clock pulses and packets propagate in the opposite directions. Assume that the propagation delays between n_1 and n_2 on the clock waveguide are d units, we have $T_L(1) = T_L(2) + d$. If the two nodes are to perform an operation simultaneously according to their local times, then the node n_1 will start d units earlier than the node n_2. Therefore, these two nodes will not perform the operation on the same packet as long as $P \leq D + d$. That is, given a fixed D, the packet size can be increased by d units using the skewed-time model. Note that, if clock pulses and packets propagate in the same direction, the packet size will have to be decreased. We are interested in increasing the packet size for a given D to overcome current technology restraints on the pulse width.

In order to have a regular connection between switches and the global clock, the skewed-time model is used to distribute the global clock to switches in *ASOS*, as shown in *Figure 11*. Clock pulses always propagate in a direction opposite to that of packet propagation. For example, packets propagate on the lower segment of a row bus from left to right. But at any row bus r, the following equation holds :

$$ST_L(r,i) = ST_L(r,i-1) + d \qquad (4.1)$$

that is, clock pulses propagate from right to left. Suppose that switch $i+1$ has just been set to the cross state when PS_{i+1} arrives. The beginning of PS_i is $\overline{D}$ units away from that of PS_{i+1} and $\overline{D} - D = d$ units away from switch i. After d units, switch i will be set to the cross state and PS_i will arrive at switch i as desired. In addition, switched packets propagate top down on the left segment of a column bus but since $ST_L(r,c) = ST_L(r-1,c) + d$, clock pulses propagate bottom up at each column. Therefore, no packet overlappings are possible on any column bus.

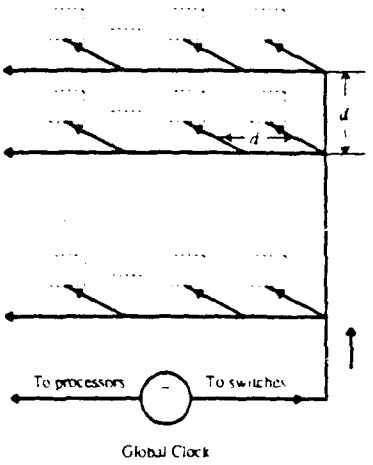


Figure 11. Control switches with skewed-time

We can similarly connect all processors to the same global clock using the skewed-time model. Note that, processors transmit their packets on the upper segment of row buses. These packets propagate from right to left at each row and therefore clock pulses should propagate from left to right at each row.

It is possible to recalculate equation (2.1) for the skewed-time model in terms of the local time of each processor so that communications and reservations can be carried out as described in Section 2 and Section 3 respectively [19]. Note that, this skewed-time model can also be applied in other time-division multiplexing or pipelining systems.

5. Concluding remarks

It has been established that much higher bandwidth can be achieved with pipelined optical bus interconnections than with electronic exclusive bus access communications [7, 16]. The high bandwidth of optical waveguides can also be achieved using two novel time-division approaches, namely TDSM and TDDM. These two approaches are used in the design presented in this paper.

In addition to achieving high bandwidth, several other design goals are met. One is the simplicity of the hardware structures and controls. Only one switch is used for each row and each column bus and switch settings are performed uniformly and synchronously. Another related goal is the feasibility and the flexibility of the design given the available technologies. For example, with current technology, it is possible to drive an optical bus at 20 GHz [22]. This results in a pulse width w, or a time unit as defined in this paper, of 50 ps (pico-second). Assuming that the speed of light in the waveguide is $c_g = 2\times10^8$ m/sec, the spatial length of a pulse is 1 *cm*. With 100 ps switching speed [22] and a 16-bit message frame, the switching time S is 2 time units and the packet length P is 16 time units in an 8×8 array. With backplane connections, it is reasonable to assume that the spatial separation of two processors, D, is 7 *cm*, or equivalently 7 time units. The condition to prevent packet overlapping, namely $D \geq P + S$, is thus not satisfied. Therefore, the skewed-time model should be used and the skewing distance d should be at least 11 time units. Note that if communication loads are 80% of the communication capacity in both row and column phases, that is, $L_r = L_c = 0.8$, the maximum effective bandwidth for the 8×8 array is about 113.8 Gb/sec according to equation (3.4).

When technology advances to the stage at which optical buses are implemented on GaAs wafers with 100 GHz transmission and at 10 ps switching speed, the skewed-time model may be no longer necessary. In fact, with $D = 7$ *cm*, a packet could contain up to 34 bits without using the skewed-time model. Nevertheless, the skewed-time model should be used if D is reduced to 1 cm (the order of the separation in chip-to-chip connections).

We note that our reservation schemes that relate to TDDM are different from any of the schemes discussed in [6]. Also, uses of random access schemes, such as those in [15, 17], for resolving time-slot contentions would yield longer average packet delays and lower system bandwidth in this particular application in which the communication load is assumed to be high. Finally, we note that some technology issues not mentioned in this

paper, such as pulse generations, coincidence detections and power distributions, are discussed in [18, 4].

Acknowledgement

This research is supported, in part, by a grant from the Air Force Office of Scientific Research under grant #AFOSR-89-0469.

References

1. R. Alferness, L. Buhl, S. Korotky, and R. Tucker, "High-Speed $\Delta\beta$-Reversal Directional Coupler Switch," *Topical Meeting on Photonic Switching, Technical Digest Series*, vol. 13, pp. 77-78, 1987.
2. A. Benner, H. Jordan, and V. Heuring, "Optically Switched Lithium Niobate Directional Couplers for Digital Optical Computing," *SPIE Proc., Digital Optical Computing II*, vol. 1215, pp. 343-352, 1990.
3. D. Chiarulli, R. Melhem, and S. Levitan, "Using Coincident Optical Pulses for Parallel Memory Addressing," *IEEE Computer*, vol. 20, no. 12, pp. 48-58, 1987.
4. D. Chiarulli, R. Ditmore, R. Melhem, and S. Levitan, "An All Optical Addressing Circuit : Experimental Results and Scalability Analysis," *IEEE Journal of Lightwave Technology, Special issue on optical interconnection for information processing*, (to appear).
5. J. Cohen, "The Single Server Queues," *North Holland*, 1969.
6. M. Fine and F. Tobagi, "Demand Assignment Multuple Access Schemes in Broadcast Bus Local Area Networks," *IEEE Transactions on Computers*, vol. C-33, no. 12, pp. 1130-1159, Dec. 1984.
7. Z. Guo, R. Melhem, R. Hall, D. Chiarulli, and S. Levitan, "Array Processors with Pipelined Optical Busses," *Journal of Parallel and Distributed Computing*, vol. 12, no. 3, pp. 269-282, 1991.
8. P. Haugen, S. Rychnovsky, A. Husain, and L. Hutcheson, "Optical Interconnects for high speed computing," *Optical Engineering*, vol. 25, pp. 1076-1085, Oct. 1989.
9. F. Kiamilev, S. Esener, V. Ozgus, and S. Lee, "Programmable Optoelectronic Multuprocessor Systems," *Digital Optical Computing, SPIE Press.*, vol. CR35, pp. 197-220, July 1990.
10. L. Kleinrock, "Queueing Systems, Volumn 1 : Theory," *John Wiley and Sons*, 1975.
11. H. Kobayashi and A. Konheim, "Queueing Models for Computer Communications Systems Analysis," *IEEE Transactions on Communications*, vol. COM-25, no. 1, pp. 2-29, Jan. 1977.
12. A. Konheim, "Chaining in a Loop System," *IEEE Transactions on Communications*, vol. COM-24, no. 2, pp. 203-210, Feb. 1976.
13. S. Levitan, D. Chiarulli, and R. Melhem, "Coincident Pulse Techniques for Multiprocessor Interconnection Structures," *Applied Optics*, vol. 29, no. 14, pp. 2024-2039, 1990.
14. J. Little, "A Proof for the Queueing Formula : $L = \lambda$," *Operations Research*, vol. 9, no. 4, pp. 204-209, July 1961.
15. N. Maxemchuk, "Twelve Random Access Strategies for Fiber-Optic Networks," *IEEE Transactions on Communications*, vol. 36, pp. 942-950, Aug. 1988.
16. R. Melhem, D. Chiarulli, and S. Levitan, "Space Multiplexing of Waveguides in Optically Interconnected Multiprocessor Systems," *The Computer Journal*, vol. 32, no. 4, pp. 362-369, 1989.
17. R. Metcalfe and D. Boggs, "Ethernet : Distributed Packet Switching for Local Computer Networks," *Communications of ACM*, vol. 19, no. 7, pp. 395-403, 1976.
18. M. Nassehi, F. Tobagi, and M. Marhic, "Fiber Optic Configurations for Local Area Networks," *IEEE Journal on Selected Areas in Communication* , vol. SAC-3, no. 6, pp. 941-949, Nov. 1985.
19. C. Qiao and R. Melhem, "Time-Division Optical Communications In Multiprocessor Arrays," *Tech. Rep. 91-14*,, CS Department, University of Pittsburgh , March 1991.
20. C. Qiao, D. Chiarulli, R. Melhem, and S. Levitan, "Optical Multicasting in Linear Arrays," *International Journal of Optical Computing*, (to appear).
21. S. Ramanan and H. Jordan, "Serial Array Shuffle-Exchange Architecture for Universal Permutation of Time Slots," *SPIE Proc., Digital Optical Computing II*, vol. 1215, pp. 330-342, Jan. 1990.
22. J. Sauer, "A Multi-Gb/s Optical Interconnect," *SPIE Proc., Digital Optical Computing II*, vol. 1215, pp. 198-207, 1990.
23. H. Takagi, "Analysis of Polling Systems," *MIT Press*, 1986.
24. R. Thompson and P. Giordano, "An Experimental Photonic Time-slot Interchanger Using Optical Fibers as Reentrant Delay-line Memories," *IEEE J. Lightwave Technology* , vol. 1, pp. 154-162, 1987.